Theoretical Computer Science 940 (2023) 18–42

Contents lists available at ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

How majority-vote crossover and estimation-of-distribution algorithms cope with fitness valleys

Carsten Witt

DTU Compute, Technical University of Denmark, 2800 Kgs. Lyngby, Denmark

ARTICLE INFO

Article history:
Received 19 December 2021
Received in revised form 24 June 2022
Accepted 15 August 2022
Available online 28 August 2022

Keywords:
Randomized search heuristics
Crossover
Estimation-of-distribution algorithms
Jump functions
Multimodal functions
Runtime analysis

ABSTRACT

The benefits of using crossover in crossing fitness gaps have been studied extensively in evolutionary computation. Recent runtime results show that majority-vote crossover is particularly efficient at optimizing the well-known Jump benchmark function that includes a fitness gap next to the global optimum. Also estimation-of-distribution algorithms (EDAs), which use an implicit crossover, are much more efficient on Jump than typical mutation-based algorithms. However, the allowed gap size for polynomial runtimes with EDAs is at most logarithmic in the problem dimension n.

In this paper, we investigate variants of the Jump function where the gap is shifted and appears in the middle of the typical search trajectory. Such gaps can still be overcome efficiently in time $O(n \log n)$ by majority-vote crossover and an estimation-of-distribution algorithm, even for gap sizes almost $\sqrt{n}$. However, if the global optimum is located in the gap instead of the usual all-ones string, majority-vote crossover would nevertheless approach the all-ones string and be highly inefficient. In sharp contrast, an EDA can still find such a shifted optimum efficiently. Thanks to a general property called *fair sampling*, the EDA will with high probability sample from almost every fitness level of the function, including levels in the gap, and sample the global optimum even though the overall search trajectory points towards the all-ones string. Finally, we derive limits on the gap size allowing efficient runtimes for the EDA.

1. Introduction

In the theory of evolutionary algorithms (EAs) there is an ongoing debate on the utility of crossover. Recent works have studied the advantage of different crossover operators in overcoming fitness gaps, in particular on the concrete example of the well-known Jump benchmark function. This function lives on the search space of bit strings of length n and separates the global optimum, located in the all-ones string, from a set of local optima by a Hamming distance of m, where $m > 1$ is a parameter. Simple mutation-based hill climbers like the well-known (1+1) EA typically have to flip m bits simultaneously to leave the set of local optima, resulting in an expected optimization time of $\Theta(n^m)$ [17]. We will discuss more advanced algorithms with mutation as only search operator in Section 2.

The Jump function was the first example where a rigorous runtime analysis showed the usefulness of crossover. Starting with the seminal paper by Jansen and Wegener [27], a series of works [29,42,8] presented refined analyses of the original setting from [27] and introduced modified algorithms that optimize Jump much more efficiently than the simple (1+1) EA. Extremely roughly speaking, the insights are: under certain assumptions, the n^m term mentioned above can be replaced

E-mail address: cawi@dtu.dk.

https://doi.org/10.1016/j.tcs.2022.08.014

with a $2^{\Theta(m)}$ term if small crossover rates are used and with $\Theta(n^{m-1})$ for larger crossover rates. All these results consider uniform crossover, i.e., an operator that uses two parents and where the offspring inherits uniformly at random from either parent.

There are also several recent works [19,44,41] that apply a majority-vote crossover to optimize Jump, i.e., an operator that uses a larger, typically odd number of parents and where the offspring inherits the most frequent bit value from the parents. This results in extremely efficient times to overcome the fitness gap of Jump. For instance, in Whitley et al. [44], majority-vote crossover leads to a total running time of $\Theta(n)$ with high probability for gaps smaller than $\log n$ since the "easy" part of the function (resembling the well-known OneMax) is done by a specific local search operator in a so-called *hybrid GA*. In Rowe and Aishwaryaprajna [41], the analysis of this hybrid GA was significantly improved, proving that it even overcomes gaps of size $o(\sqrt{n})$ in $\Theta(n)$ time with high probability. Moreover, Rowe and Aishwaryaprajna [41] proposed a different algorithm based on majority decisions called the *voting algorithm* that solves Jump efficiently, more precisely in $O(n \log n)$ fitness evaluations with high probability, for gap sizes up to $(1-\epsilon)n/2$ for an arbitrarily small constant $\epsilon > 0$. Majority-vote crossover was also investigated in Friedrich et al. [19]. That work considers, amongst other algorithms, the so-called single receiver island model of evolutionary algorithms, which, similar to the above-mentioned hybrid GA, can optimize Jump up to gap sizes $o(\sqrt{n})$ efficiently in $O(n \log n)$ fitness evaluations with high probability.

Recently, there has also been interest in the runtime analysis of estimation-of-distribution algorithms (EDAs) on the Jump function. Doerr [10] has shown that a simple estimation-of-distribution algorithm called cGA optimizes Jump in time roughly $O(n \log n + 2^{O(m)})$ with high probability, hence $O(n \log n)$ if $m \leq c \ln n$ for a sufficiently small constant $c > 0$. Since the cGA is a model of a crossover-based algorithm, it is also relevant for our discussion of the utility of crossover for overcoming fitness gaps. Also very recently [5], a simple ACO algorithm that shares many features with the cGA was analyzed on Jump and a result similar to Doerr [10] was shown in that the ACO algorithm can overcome gap sizes $m \leq c \ln n$ for a sufficiently small constant $c > 0$ efficiently.

A common trait of all these algorithms seems to be that they "learn" for each bit position that a one-entry is more beneficial than a zero and implicitly are led towards the all-ones string. Whitley et al. [44] describe this phenomenon as exploiting "first order" information, i.e., setting a bit to one at an arbitrary position yields higher fitness than setting it to zero, on average over all outcomes of the remaining $n-1$ bits and even for any fixed outcome of these bits. Doerr [10] points out that the cGA can overcome fitness gaps thanks to a higher sampling variance compared to simple mutation-based EAs. At the same time, one can say that the probabilistic sampling mechanism in the cGA learns for every bit position that a one-entry is more beneficial than a zero.

We shall investigate whether the above-mentioned benefits of crossover-based algorithms on Jump still exist if the global optimum is far away from the all-ones string. In fact, we aim at pointing out situations where benefits cease to exist if the optimum is located elsewhere. To this end, we present a simple situation where the crossover, through an implicit majority decision, is lead to the all-ones string while overcoming a large fitness gap. However, the all-ones string is only a local optimum. It is not too difficult to show that several of the above-mentioned crossover-based algorithms are highly inefficient in this situation.

However, our study will be more comprehensive than this simple negative result for crossover-based algorithms. In fact, we will start our paper with a discussion of the location of the gap in the Jump function and propose a shift of the gap towards the middle of the typical search trajectory. Independently, generalized Jump functions of this kind were proposed in the very recent work by Bambury et al. [4], where the efficiency of mutation-based EAs is studied. Here we analyze the simple crossover-based algorithms and the cGA. In a nutshell, the shift of the gap allows the cGA to overcome gaps of size roughly $\sqrt{n}$, and also the other algorithms can overcome this gap efficiently. However, the crossover-based algorithms will produce search points with a high number of one-bits. In particular, if there is a global optimum hidden in the middle of the gap, they will with high probability not produce such a point and need superpolynomial time to find the global optimum. Surprisingly, in this case, the cGA can still be highly efficient. In a separate section of our paper (Section 4), we establish a property of the cGA related to fair sampling on Hamming levels while following a fitness gradient, akin to the famous OneMax landscape. This highlights a fundamental difference between the crossover-based algorithms using majority-vote crossover and the cGA.

We will round off our study by analyzing more precisely the largest gap size that the cGA can overcome in polynomial time. Here we find that gap size asymptotically larger than $\sqrt{n} \log^2 n$ cannot be crossed efficiently with high probability, at least if a slightly simplified cGA is used.

This paper is structured as follows. In Section 2, we introduce the algorithms and functions studied in the paper and collect important mathematical tools for the analysis. In Section 3, we consider the simple case of a jump function with shifted gap and show mostly positive results for both the crossover-based EAs and the cGA. Section 4 establishes the fair sampling property of the cGA. This is used in Section 5 to show that the cGA is efficient on a jump function with shifted optimum and shifted gap, while crossover-based algorithms are highly inefficient. Section 6 shows a limit on the gap size that can be overcome with a variant of the cGA. In Section 7 we show some experimental studies, supporting our asymptotic results also for small problem dimensions. We finish with some conclusions.

Extensions compared to conference paper A preliminary version of this work appeared in the proceedings of FOGA '21 [47]. This article extends the conference version with full proofs and experimental supplements.

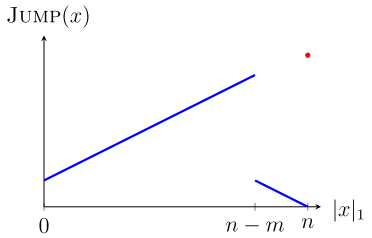

Fig. 1. Illustration of JUMP; fitness gap between $n-m$ and n one-bits and optimum at n one-bits.

2. Preliminaries

2.1. Jump functions and their modifications

The JUMP_m function with so-called gap size m, introduced in Droste et al. [16], is a well-known pseudo-Boolean benchmark function designed to analyze the ability of randomized search heuristics to overcome local optima. It has a fitness gradient corresponding to the number of one-bits of the search point up to the point of $n-m$ one-bits. Search points having between $n-m+1$ and $n-1$ one-bits are inferior than all other search points (which is called a fitness gap), with a gradient pointing towards the set of search points with $n-m$ one-bits. Finally, the all-ones string is the global optimum. Denoting by $|x|_1$ the number of one-bits in a search point $x \in \{0,1\}^n$, we formally define for $m \in \{1,\dots,n\}$

$$\text{JUMP}_m(x) = \begin{cases} m + |x|_1 & \text{if } |x|_1 \leq n-m \text{ or } |x|_1 = n, \\ n - |x|_1 & \text{otherwise.} \end{cases}$$

See Figure 1 for an illustration. Hence, for $m=1$, JUMP equals $|x|_1+1$ and is isomorphic to the famous function $\text{ONEMAX}(x) = |x|_1$. If $m=n$, it is isomorphic to the fully deceptive TRAP function [1].

As already mentioned in the introduction, the JUMP function is a test bed for the study of randomized search heuristics on multimodal problems and has proved as an example for the benefits of crossover-based algorithms compared to mutation-only algorithms. A simple hill-climber like the famous (1+1) EA (see Algorithm 4) has expected optimization time $\Theta(n^m)$ on JUMP_m for $m \geq 2$ [16] since it with overwhelming probability will reach a local optimum at $n-m$ one-bits and then has to flip m bits simultaneously to make the final improving step. However, the mutation probability $1/n$ used in the (1+1) EA makes this m-bit flip rather unlikely. Using the optimal mutation probability m/n – which requires knowledge of the gap size – the optimization time is reduced to $O((en/m)^m)$. This time can be achieved or almost achieved with mutation-only algorithms that use heavy-tailed or self-adjusting mutation schemes [13,21,38]. If a self-adjusting mutation scheme flipping exactly k out of n bits is used and k is adjusted over time, the optimization time is upper bounded by $O(\binom{n}{m})$ [40]. Similarly, artificial immune systems with specialized mutation operators come close to this bound, e.g., the (1+1) IA^{hyp} [6] with a runtime of $O(n\binom{n}{m})$ and the (1+1) FAST-IA [7] with a runtime of $O(m(\log n)\binom{n}{m})$. Finally, the (1+(λ,λ)) GA with carefully chosen parameters optimizes JUMP_m in expected time $O(n^{(m+1)/2}e^{O(m)}m^{-m/2})$ [2]. Still, all these bounds are essentially no better than $n^{m/2}/m!$ and therefore exponentially growing with respect to m (if $m = o(n)$). This is in sharp contrast to the results for the crossover-based EAs (in particular those with majority-vote crossover) and the simple EDA cGA mentioned in the introduction.

As motivated above, the aim of this work is to study variants of JUMP where

(a) the fitness gap is not directly adjacent to the all-ones string,
(b) the optimum is not located at the all-ones string,
(c) and a combination of both.

Type (a) of modification gives rise to a class of generalized jump functions with two parameters: the start of the gap k and its size m. The function then gives a fitness value of $|x|_1 + m$ to all search points having at most k and at least $k+m$ one-bits, while search points with a number of ones in $(k, k+m)$ have a fitness of $k+m-|x|_1$. If $k = n-m$, we recover the original JUMP_m function; however, for smaller k there is not only a single point after the gap (the all-ones string) but $\binom{n}{k+m}$ many points, which may increase the probability of crossing the gap with mutation-based algorithms (see [4]).

It is clear that choices of k below $n/2-m$ are not very interesting since a uniform search point would then have a good probability of being behind the gap. Since we are mostly interested in the size of the gap that can be overcome and would like to avoid another parameter, we fix $k = 3n/4$ in this work, i.e., the gap starts half-way between the expected starting point of a random bit string and the global optimum, the all-ones string, and therefore assume $m < n/4$. Most of the results of this paper can be easily generalized to the case that k has linear distance from both $n/2$ and n, i.e., $k \in [(1+\epsilon)(n/2), (1-\epsilon)n]$ for some constant $\epsilon > 0$.

With the start of the gap fixed at $3n/4$, we obtain our function called JUMPOFFSET for "Jump with offset", see Figure 2 for an illustration.

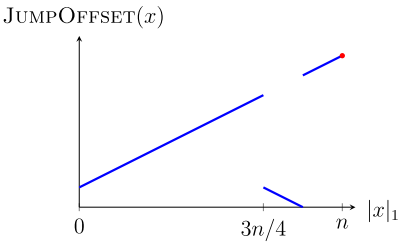

Fig. 2. Illustration of JumpOffset; fitness gap between $(3/4)n$ and $(3/4)n+m$ one-bits and optimum at n one-bits.

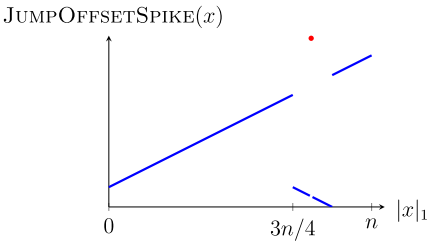

Fig. 3. Illustration of JumpOffsetSpike; fitness gap between $(3/4)n$ and $(3/4)n+m$ one-bits except for the optima at $(3/4)n+m/2$ one-bits.

$$\textsc{JumpOffset}_m(x) = \begin{cases} m + |x|_1 & \text{if } |x|_1 \leq 3n/4 \text{ or } |x|_1 \geq 3n/4 + m, \\ 3n/4 + m - |x|_1 & \text{otherwise.} \end{cases}$$

This function (with a variable start of the gap, i.e., with the two parameters k and m mentioned above) was introduced by Rajabi and Witt [39] to investigate whether self-adjusting, mutation-only algorithm with a memory mechanism would be able to first increase its mutation strength to overcome the gap and to quickly decrease it again afterwards. Independently, this generalization of Jump was proposed in the very recent work by Bambury et al. [4] to investigate how quickly various mutation-based algorithms overcome a gap that is not adjacent to the global optimum but starts at some value $k < n$, which is assumed as $n - o(n^{1/3})$ when analyzing the (1+1) EA. Crucially different from the present work, we consider a smaller k (fixed at $3n/4$ in the remainder) and focus on crossover-based algorithms, including the cGA.

The modification (b) mentioned above was essentially the key idea for the so-called RealJump function proposed in Jansen [26]. That function equals the original (non-shifted) $\textsc{Jump}_m$ function except for the fact that the global optimum is a single search point x^* such that $|x^*|_1 > n - m$. Hence, in a sense, the gap region involves a needle-in-a-haystack problem of size $\binom{n}{m}$, which results in a black-box complexity proportional to the number of search points in the gap region, formally analyzed for constant m in Jansen [26]. Due to this needle property, we do not expect very interesting performance differences between the crossover-based EAs and the EDA studied in this paper and do not consider the RealJump function further.

The third modification, type (c) from above, will illustrate a situation where the cGA displays a very different search behavior compared to the crossover-based algorithms. We take the JumpOffset function and move the global optimum to a Hamming level (not a single point) in the gap. Hence, there is a "spike" in the gap region which we place at $3n/4 + m/2$, i.e., in the middle of the gap, to avoid introducing another parameter. Moreover, we assume $m \geq 4$ for this function to make sure that the neighboring Hamming levels of the spike are inferior. We call the resulting function JumpOffsetSpike (see Figure 3) and define it as follows:

$$\textsc{JumpOffsetSpike}_m(x) = \begin{cases} m + |x|_1 & \text{if } |x|_1 \leq 3n/4 \text{ or } |x|_1 \geq 3n/4 + m, \\ n + m + 1 & \text{if } |x|_1 = 3n/4 + m/2, \\ 3n/4 + m - |x|_1 & \text{otherwise.} \end{cases}$$

We emphasize that the set of global optima of JumpOffsetSpike is the complete Hamming level of $3n/4 + m/2$ one-bits; in Section 5 we will briefly discuss variants where the global optima are more sparse. As we will see, the different location of the global optima makes the function hard to optimize for the different crossover-based algorithms considered in this work, whereas the simple estimation-of-distribution algorithm cGA is efficient for all gap sizes $m = O(\sqrt{n}/\log^5 n)$. More generally speaking, JumpOffsetSpike demonstrates how different search heuristics explore the search space when they are stuck in a local optimum.

2.2. Algorithms

We now formally introduce the different crossover-based algorithms studied in this paper. They all use *majority-vote crossover* that takes a multi-set of parents $x^{(1)}, \dots, x^{(\mu)} \in \{0,1\}^n$, where μ is usually an odd integer to avoid ties, as input and produces the offspring $z \in \{0,1\}^n$ by setting bit z_i, where $i \in \{1,\dots,n\}$, to the most frequent bit value in the multi-set; formally, $z_i := \lfloor (\sum_{j=1}^{\mu} x_i^{(j)}/\mu) + 1/2 \rfloor$.

The first algorithm we consider, called the *hybrid GA*, was introduced by Whitley et al. [44] to present a randomized search heuristic that can optimize the classical $\textsc{Jump}_m$ function in linear time $O(n)$ for gap sizes up to $\lfloor \log n \rfloor$; as already mentioned this actually holds up to gap sizes of $O(n^{1/2-\epsilon})$ for an arbitrarily small constant $\epsilon > 0$ thanks to the improved analysis in [41]. Note that there is not really a single, concise algorithm description of the hybrid GA in Whitley et al. [44]; however, it seems that the hybrid GA improves all members of the current population through next-ascent hillclimbing (using one-bit flips following a random permutation of the indices) until a local optimum is found. Thereafter, majority-vote crossover is applied and the offspring returned as the final result. The population size considered in concrete analyses equals 3; however, generalizations to arbitrary population sizes, as discussed in Whitley et al. [44], are obvious. We give our formulation of the hybrid GA as pseudo code in Algorithm 1; throughout this paper, we assume that the underlying pseudo-Boolean function $f : \{0,1\}^n \to \mathbb{R}$ given as input to an algorithm is to be maximized.

Algorithm 1: The Hybrid GA from Whitley et al. [44].

```
for k = 1, 2, 3 do
    Let x^(k) ∈ {0,1}^n be uniformly at random;
    repeat
        Let σ be a permutation on {1,...,n} drawn u.a.r.;
        for i = 1,...,n do
            Create y by flipping bit position σ(i) of x^(k);
            if f(y) > f(x^(k)) then x^(k) ← y;
    until none of the n bit flips resulted in f(y) > f(x^(k));
Create z by applying majority-vote crossover with parents x^(1), x^(2), x^(3) and query f(z);
```

The second algorithm is called the *voting algorithm* and was introduced by Rowe and Aishwaryaprajna [41]. It does not involve mutation, but runs μ tournaments comparing the fitness of two search points drawn uniformly at random. The μ winners of the tournaments then undergo majority-vote crossover and the offspring is returned as result. See the pseudo code in Algorithm 2. As a minor detail, we note that majority-vote crossover as defined in Rowe and Aishwaryaprajna [41] only requires a soft majority, whereas the hybrid GA as defined in Whitley et al. [44] requires a strict majority; however, this has no relevance for the results proved later in this paper.

Algorithm 2: The Voting Algorithm [41].

```
Let p = (0,...,0);
repeat
    Let x ∈ {0,1}^n be uniformly at random;
    Let y ∈ {0,1}^n be uniformly at random;
    if f(x) > f(y) then
        p = p + x;
    else
        p = p + y;
until μ times;
for i = 1,...,n do
    if p_i > μ/2 then
        z_i = 1;
    else
        z_i = 0;
query f(z);
```

As already mentioned, there are further studies of majority-vote crossover in the context of the classical $\textsc{Jump}_m$ function. Most importantly, the single receiver island model from Friedrich et al. [19] can optimize $\textsc{Jump}$ up to gap sizes $o(\sqrt{n})$ efficiently. The underlying effect is similar to the hybrid GA. Hence, for the sake of simplicity, we do not go into detail with the single-receiver island algorithm in this paper. However, we believe that most negative results proven for the hybrid GA in Section 5 also extend to the single-receiver island algorithm straightforwardly.

We now turn to the simple estimation-of-distribution algorithm that has rather different search dynamics to the above crossover-based algorithms but is still very efficient on the classical $\textsc{Jump}$ function up to gap sizes $m \leq c \ln n$ for a sufficiently small constant $c > 0$. The algorithm is called the *compact Genetic Algorithm (cGA)* and was introduced in Harik et al. [23]. It evolves a so-called frequency vector $\boldsymbol{p}_t = (p_{t,1}, \dots, p_{t,n})$ over time to maximize $f : \{0,1\}^n \to \mathbb{R}$. At iteration t, two bit

strings, so-called individuals or offspring, are sampled by letting frequency $p_{t,i}$ be the marginal probability of setting bit i to 1. Afterwards, the two individuals are ranked by fitness and a frequency is changed if the two individuals differ in the bit; more specifically, if the bit is 1 in the better offspring, the frequency is increased by $1/K$, otherwise decreased by $1/K$. Here $K \geq 2$, the so-called (hypothetical) population size, is a real-valued parameter to the algorithm that crucially determines its runtime [15,20,43]. Finally, all frequencies are restricted to the interval $[1/n, 1-1/n]$, whose limits are called *borders* for the frequencies, to avoid frequencies be stuck irreversibly at 0 or 1. See Algorithm 3 for a description in pseudo code.

Algorithm 3: Compact Genetic Algorithm (cGA).

```
t ← 0;
p_{t,1} ← p_{t,2} ← ··· ← p_{t,n} ← 1/2;
while termination criterion not met do
    for i ∈ {1, ..., n} do
        x_i ← 1 with prob. p_{t,i}, x_i ← 0 with prob. 1 − p_{t,i};
    for i ∈ {1, ..., n} do
        y_i ← 1 with prob. p_{t,i}, y_i ← 0 with prob. 1 − p_{t,i};
    if f(x) < f(y) then swap x and y;
    for i ∈ {1, ..., n} do
        if x_i > y_i then p_{t+1,i} ← p_{t,i} + 1/K;
        if x_i < y_i then p_{t+1,i} ← p_{t,i} − 1/K;
        if x_i = y_i then p_{t+1,i} ← p_{t,i};
        p_{t+1,i} ← max{min{p_{t+1,i}, 1 − 1/n}, 1/n};
    t ← t + 1;
```

Finally, for the sake of completeness, we formally introduce the well-known (1+1) EA, which is not a crossover-based algorithm, but a simple hillclimber using independent bit-flip mutation, see Algorithm 4. It was the first algorithm whose runtime on Jump was analyzed. Crucially, the expected runtime of the (1+1) EA on $\textsc{Jump}_m$ is $\Omega(n^m)$ [16], i.e., grows exponentially in m.

Algorithm 4: (1+1) EA.

```
t ← 0; select x_0 uniformly at random from {0, 1}^n;
while termination criterion not met do
    Create y by flipping each bit of x_t independently with probability 1/n;
    if f(y) ≥ f(x_t) then x_{t+1} ← y;
    else x_{t+1} = x_t;
    t ← t + 1;
```

For all algorithms, we denote the *runtime* as the number of f-evaluations until an optimal search point of f is generated. With respect to the cGA and (1+1) EA, this is proportional to the number of iterations until an optimal search point is sampled. For the hybrid GA, it is at least $3(n+1)+1 = 3n+4$, and for the voting algorithm it is always $2\mu+1$.

2.3. General probabilistic tools

In this subsection, we collect several technical tools related to drift, concentration and anti-concentration that we use in our upcoming runtime analyses, in particular when investigating the cGA. However, these tools are formulated independently of concrete algorithms, whereas the next subsection deals with more specific results related to the cGA. The reader may skip this and the following subsection at first reading and come back to it when studying the proofs of our main results.

The following lemma dealing with Chernoff-type bounds depending on the variance goes back to Hoeffding [24]. We use a bound given in Doerr [11].

Lemma 1 *(Th. 1.10.14 in [11]). Let $X_1, \dots, X_n$ be independent random variables. Let b be such that $X_i \leq \mathrm{E}(X_i) + b$ for all $i = 1, \dots, n$. Let $X = \sum_{i=1}^n X_i$. Let $\sigma^2 = \sum_{i=1}^n \mathrm{Var}(X_i) = \mathrm{Var}(X)$. Then, for all $\lambda \geq 0$,*

$$\Pr(X \geq \mathrm{E}(X) + \lambda) \leq e^{-(1/3)\min\{\lambda^2/\sigma^2, \lambda/b\}}.$$

A random variable follows the Poisson-binomial distribution if it is the sum of n independent Poisson trials (0-1 random variables) with individual success probabilities $p_1, \dots, p_n$. The Poisson-binomial distribution arises naturally in the analysis of the number of one-bits sampled by the cGA. The next three lemmas, i.e., Lemmas 2–4, all consider the Poisson-binomial distribution. Our presentation of the lemmas follows closely the original sources and varies therefore slightly in notation.

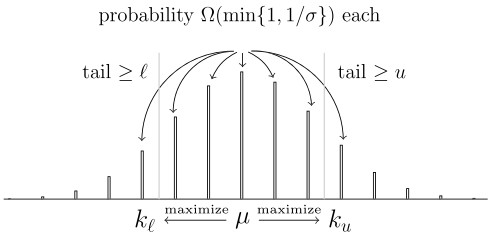

Fig. 4. Illustration of Lemma 4.

The first lemma is concerned with anti-concentration and bounds the probability function of the Poisson-binomial distribution uniformly. The constant $1/2$ appearing there can be improved to $\eta < 0.4689$ [3]; however, this is not relevant for our application.

Lemma 2 *([3]). Let X follow a Poisson-binomial distribution with parameters $p_1, \dots, p_n$ and let $\sigma^2 = \mathrm{Var}(X)$. Then for all $k \in \{0, \dots, n\}$ it holds that*

$$\Pr(X = k) \leq \frac{1}{2\sigma}.$$

The next lemma shows that a random variable following a Poisson-binomial distribution deviates from its expectation by at least roughly the variance with at least a constant probability. It was proven in Doerr [11] using a clever two-stage rounding trick.

Lemma 3 *(Th. 1.10.16 in [11]). Let $v_0 \geq 0$. There are constants $c_1, c_2 > 0$ such that the following is true. Let $n \in \mathbb{N}$. Let $p_1, \dots, p_n \in [0, 1]$. For all $i \in \{1, \dots, n\}$, let X_i be a binary random variable with $\Pr(X_i = 1) = p_i$. Assume that $X_1, \dots, X_n$ are independent. Let $X = \sum_{i=1}^n X_i$. Assume that $\mathrm{Var}(X) = \sum_{i=1}^n p_i(1 - p_i) \geq v_0$. Then*

$$\Pr\big(X \geq \mathrm{E}(X) + c_1\sqrt{v_0}\big) \geq c_2,$$
$$\Pr\big(X \leq \mathrm{E}(X) - c_1\sqrt{v_0}\big) \geq c_2.$$

The following lemma from Witt [46] considers a chunk of the Poisson-binomial distribution around the expected value such that the joint probability of the chunk is a constant less than 1 and then argues that every point in the chunk has a probability that is at least inversely proportional to the variance. See Figure 4 for an illustration (taken from [46]). To clarify the use of Ω-notation in the last sentence of the lemma, we remark that the statement means that for large enough n there is a constant $c > 0$ such that $\Pr(X = k) \geq c \min\{1, 1/\sigma\}$.

Lemma 4 *(Lemma 2 in [46]). Let $X_1, \dots, X_n$ be independent Poisson trials. Denote $p_i = \Pr(X_i = 1)$ for $i \in \{1, \dots, n\}$, $X := \sum_{i=1}^n X_i$, $\mu := \mathrm{E}(X) = \sum_{i=1}^n p_i$ and $\sigma^2 := \mathrm{Var}(X) = \sum_{i=1}^n p_i(1 - p_i)$. Given two constants $\ell, u \in (0, 1)$ such that $\ell + u < 1$, let $k_\ell := \min\{i \mid \Pr(X \leq i) \geq \ell\}$ and $k_u := \max\{i \mid \Pr(X \geq i) \geq u\}$. Then it holds that $\Pr(X = k) = \Omega(\min\{1, 1/\sigma\})$ for all $k \in \{k_\ell, \dots, k_u\}$, where the Ω-notation is with respect to n.*

When estimating the accumulated progress of the cGA in k steps, we are pessimistically confronted with the sum of k random variables, each of which is distributed as the absolute value of the total bit-wise difference of two bit strings drawn from the current sampling distribution. We prove a simple, non-asymptotic result based on Hoeffding's inequality but note that similar bounds could also be obtained via the Normal approximation.

Lemma 5. *Let $k, n \in \mathbb{N}$. Let $Z_1, \dots, Z_k$ be random variables such that for each $i \in \{1, \dots, n\}$ it holds that $Z_i = |\sum_{j=1}^n X_j^{(i)}|$, where each $X_j^{(i)}$ has support $\{-1, 0, 1\}$, satisfies $\mathrm{E}\big(X_j^{(i)}\big) = 0$ and is independent of all other $X_\ell^{(i)}$, where $\ell \neq j$. Then it holds for $Z := \sum_{i=1}^k Z_i$ and any $\lambda > 0$ that*

$$\Pr\big(Z \geq k\sqrt{n} + \lambda\big) \leq e^{-\lambda^2/(2kn\log^2 n)} + 2ke^{-(\log^2 n)/2}.$$

Proof. For an arbitrary but fixed i, we first analyze Z_i, which is the absolute value of the sum $S_i := \sum_{j=1}^n X_j^{(i)}$. By assumption, $\mathrm{E}\big(X_j^{(i)}\big) = 0$ for all $j \in \{1, \dots, n\}$, and by linearity of expectation, $\mathrm{E}(S_i) = 0$. Moreover, by Jensen's inequality, $\mathrm{E}(Z_i) = \mathrm{E}(|S_i|) \leq \sqrt{\mathrm{E}\big(|S_i|^2\big)} = \sqrt{\mathrm{E}\big(S_i^2\big)}$, and, since $\mathrm{Var}(S_i) = \mathrm{E}\big(S_i^2\big) - (\mathrm{E}(S_i))^2 = \mathrm{E}\big(S_i^2\big)$, we obtain $\mathrm{E}(Z_i) \leq \sqrt{\mathrm{Var}(S_i)}$. Now,

since $\mathrm{Var}(X_j^{(i)}) \leq 1$ by assumption on support and expectation of $X_j^{(i)}$, independence gives us $\mathrm{Var}(S_i) \leq n$, so altogether $\mathrm{E}(Z_i) \leq \sqrt{n}$. Note that this piece of analysis can essentially be found in Droste [15] as well.

We now use Hoeffding's inequality (e.g., Th. 1.10.9 in [11]) to bound the tail of Z_i by $\sqrt{n}\log n$, more precisely

$$\Pr\big(Z_i \geq \sqrt{n}\log n\big) \leq \Pr\big(S_i \geq \sqrt{n}\log n\big) + \Pr\big(S_i \leq -\sqrt{n}\log n\big) \leq 2e^{-2(\sqrt{n}\log n)^2/(4n)} = 2e^{-(\log^2 n)/2},$$

where the 4 in the denominator stems from the fact that the support of each $X_j^{(i)}$ is an interval of length 2.

In the following, we condition on the event G that every Z_i is bounded from above by $\sqrt{n}\log n$. Combining the last inequality with a union bound, we have that G fails to occur with probability at most $2ke^{-(\log^2 n)/2}$. Clearly, conditional on G, the expectation of every Z_i is still bounded from above by $\sqrt{n}$. Since the Z_i are not assumed independent, we cannot apply classical Hoeffding bounds on their sum; however, the independent trials behind the $X_j^{(i)}$ allow us to work with concentration inequalities for supermartingales and to obtain essentially the same result as we would have gotten with the classical Hoeffding bound for sums of independent random variables.

We define the random variables $M_i \coloneqq \sum_{j=1}^{i} Z_j - i\sqrt{n}$ and note that $(M_i)_{i=1,\dots,k}$, is a supermartingale with martingale differences in the interval $[-\sqrt{n}, \sqrt{n}\log n - \sqrt{n}]$ thanks to our condition G. Rescaling the supermartingale with a factor $1/(\sqrt{n}\log n)$ to confine the differences to $[-1/\log n, 1]$ and applying a Hoeffding bound for supermartingales [18, Corollary 2.1, Inequality (20)] with $b = 1/\log n \leq 1$ and $U_k(x,b) \leq k(1+b)^2 \leq 4k$, we have

$$\Pr\left(\max_{i=1,\dots,k} M_i \geq \lambda \mid G\right) \leq e^{-2\lambda^2/(4kn\log^2 n)} = e^{-\lambda^2/(2kn\log^2 n)},$$

which implies for $Z = \sum_{i=1}^{k} Z_i$, where $Z = \sum_{i=1}^{k} M_i + k\sqrt{n}$, that

$$\Pr\big(Z \geq k\sqrt{n} + \lambda \mid G\big) \leq e^{-\lambda^2/(2kn\log^2 n)}.$$

Plugging in our bound on $\mathrm{E}(Z \mid G)$ and adding the bound on the probability of G failing to happen, we have proved the lemma. □

The concrete application of Lemma 5 in analyzing the cGA is given in Corollary 1 in the following subsection.

The following negative drift theorem goes back to Oliveto and Witt [36] and deals with scenarios where the expected one-step change is not constant but depends on the problem size. In turn, this allows larger jumps than in older so-called simplified drift theorems. It has been applied several times to the analysis of population-based EAs [33] and also EDAs [46]. Using a straightforward generalization, we formulate the theorem for arbitrary filtrations $(\mathcal{F}_t)_{t\in\mathbb{N}}$ instead of only the natural filtration stated in Oliveto and Witt [36]. The notation $\mathrm{E}(X; A \mid \mathcal{F})$ for a random variable X, a σ-algebra $\mathcal{F}$ and an event A appearing in the theorem is defined as $\mathrm{E}(X \cdot \mathbb{1}_A \mid \mathcal{F})$; analogously for probabilities $\Pr(B; A \mid \mathcal{F})$ where B is an event.

Theorem 1 *(Negative Drift with Scaling, cf. [36]). Let $(X_t)_{t\geq 0}$ be a stochastic process, adapted to a filtration $\mathcal{F}_t$, over some state space $S \subseteq \mathbb{R}$. Suppose there exist an interval $[a,b] \subseteq \mathbb{R}$ and, possibly depending on $\ell \coloneqq b-a$, a drift bound $\epsilon \coloneqq \epsilon(\ell) > 0$ as well as a scaling factor $r \coloneqq r(\ell) > 0$ such that for all $t \geq 0$ the following three conditions hold:*

(a) $\mathrm{E}(X_{t+1} - X_t - \epsilon\,;\, a < X_t < b \mid \mathcal{F}_t) \geq 0$,
(b) $\Pr(|X_{t+1} - X_t| \geq jr\,;\, a < X_t \mid \mathcal{F}_t) \leq e^{-j}$ *for* $j \in \mathbb{N}_0$,
(c) $1 \leq r^2 \leq \epsilon\ell/(132\log(r/\epsilon))$.

Then for the first hitting time $T^* \coloneqq \min\{t \geq 0 \colon X_t \leq a \mid X_0 \geq b\}$ it holds that $\Pr\Big(T^* \leq e^{\epsilon\ell/(132r^2)} \mid \mathcal{F}_0\Big) = O\big(e^{-\epsilon\ell/(132r^2)}\big)$.

Moreover, we shall use variable drift analysis. The first theorems stating upper bounds on the hitting time of stochastic processes with variable drift go back to [28] and [35]. In this paper, we use the following recent formulation of a variable drift theorem from Hwang and Witt [25].

Theorem 2 *(Variable Drift, upper bound). Let $(X_t)_{t\geq 0}$ be a stochastic process, adapted to a filtration $\mathcal{F}_t$, over some state space $S \subseteq \{0\} \cup \mathbb{R}_{\geq x_{\min}}$, where $x_{\min} > 0$. Assume $0 \in S$ and define $T \coloneqq \min\{t \mid X_t = 0\}$.*

Let $h \colon \mathbb{R}_{\geq x_{\min}} \to \mathbb{R}^+$ be a monotone increasing function and suppose that $\mathrm{E}(X_t - X_{t+1} \mid \mathcal{F}_t) \geq h(X_t)$ conditioned on $t < T$. Then it holds that

$$\mathrm{E}(T \mid \mathcal{F}_0) \leq \frac{x_{\min}}{h(x_{\min})} + \int_{x_{\min}}^{X_0} \frac{1}{h(x)}\,\mathrm{d}x\ .$$

While the previous theorem bounds expected hitting times in the presence of variable drift, the following one bounds the tail of the hitting time in such a scenario.

Theorem 3 *(Th. 3.2.(i) in [30]). Let $(X_t)_{t\in\mathbb{N}_0}$, be a stochastic process, adapted to a filtration $\mathcal{F}_t$, over some state space $S \subseteq \{0\} \cup \mathbb{R}_{\geq x_{\min}}$, where $x_{\min} \geq 0$. Let $h\colon \mathbb{R}_{\geq x_{\min}} \to \mathbb{R}^+$ be a function such that $1/h(x)$ is integrable on $\mathbb{R}_{\geq x_{\min}}$. Suppose there exist a random variable Z and some $\lambda > 0$ such that $\left|\int_{X_{t+1}}^{X_t} 1/h(x)\,\mathrm{d}x\right| \prec Z$ for $X_t \geq x_{\min}$ for all $t \in \mathbb{N}_0$, and $E(e^{\lambda Z}) = D$ for some $D > 0$.*

Suppose that

$$\mathrm{E}(X_t - X_{t+1} - h(X_t)\,;\ X_t \geq x_{\min} \mid \mathcal{F}_t) \geq 0$$

for all $t \in \mathbb{N}_0$. Then for any $\delta > 0$, and $\eta := \min\{\lambda, \delta\lambda^2/(D-1-\lambda)\}$ and any $t^* > 0$, it holds for the first hitting time $T := \min\{t \mid X_t = 0\}$ that

$$\Pr(T > t^* \mid \mathcal{F}_0) \leq \exp\left(\eta\left(\frac{x_{\min}}{h(x_{\min})} + \int_{x_{\min}}^{X_0} \frac{1}{h(x)}\,\mathrm{d}x - (1-\delta)t^*\right)\right).$$

In the analysis of the hybrid GA (Algorithm 1), we shall apply Chernoff bounds on the number of one-bits in an individual generated by majority-vote crossover. Even though the bit positions of such an individual are not independent, the Chernoff bounds hold thanks to negatively correlated events (see Th. 1.10.23 and 1.10.24 in [11]). A sequence of random variables $X_1, \dots, X_n$ with support $\{0,1\}$ is called 0-*negatively correlated* if for all $I \subseteq \{1,\dots,n\}$ it holds $\Pr(\cap_{i\in I}\{X_i = 0\}) \leq \prod_{i\in I}\Pr(X_i = 0)$ and analogously for 1-negatively correlated random variables.

Lemma 6. *Let $n \in \mathbb{N}$, $k \in \{0,\dots,n\}$, and let $x^{(1)}, x^{(2)}, x^{(3)} \in \{0,1\}^n$ be drawn uniformly at random and independently from the set of bit strings having k one-bits. Let z be the offspring of majority-vote crossover on the parents $x^{(1)}, x^{(2)}, x^{(3)}$, and denote by the random variable Z_i its bit value at position i, $i \in \{1,\dots,n\}$. Then the random variables Z_i are both 0-negatively correlated and 1-negatively correlated.*

Proof. Let $I \subseteq \{1,\dots,n\}$ be arbitrary but fixed. The aim is to show that $\Pr(\cap_{i\in I}\{Z_i = 1\}) \leq \prod_{i\in I}\Pr(Z_i = 1)$ and analogously for 0-negative correlation. Due to the uniform distribution of the parents on the Hamming level, we can reorder the bit positions and assume hereinafter without loss of generality that $I = \{1,\dots,\ell\}$ for some $\ell \in \{1,\dots,n\}$.

By $X_i^{(1)}$, $X_i^{(2)}$ and $X_i^{(3)}$, where $i \in \{1,\dots,\ell\}$, we denote the i-th bit value of the respective parent $x^{(1)}$, $x^{(2)}$ and $x^{(3)}$. Since the three parents follow the same distribution and are independently drawn, we most of the time only consider one arbitrary but fixed of the parents and denote its bit values simply by X_i in the following. We have

$$\Pr(Z_i = 1) = (\Pr(X_i = 1))^3 + 3\cdot(\Pr(X_i = 1))^2(1 - \Pr(X_i = 1)) \tag{1}$$

since $Z_i = 1$ if and only if at least two parents have a one-bit at position i. Also, $\Pr(X_i = 1) = k/n$ for all $i \in \{1,\dots,n\}$ (without independence), and, conditioning on an outcome of $X_1,\dots,X_{i-1}$,

$$\Pr(X_i = 1 \mid X_1,\dots,X_{i-1}) = \frac{k - X_1 - \cdots - X_{i-1}}{n - i + 1}$$

since $k - X_1 - \cdots - X_{i-1}$ one-bits are uniformly distributed on the last $n - i + 1$ bits. Hence, by the law of total probability, the unconditional probability of $X_i = 1$ can be written as

$$\Pr(X_i = 1) = \frac{k - \mathrm{E}(X_1) - \cdots - \mathrm{E}(X_{i-1})}{n - i + 1}. \tag{2}$$

In the following, when considering the set $I = \{1,\dots,\ell\}$, we will condition on certain distributions of the $X_1,\dots,X_{i-1}$, where $i \leq \ell$, to estimate the joint probabilities of the Z_i.

Using the definition of conditional probability, we write

$$\Pr\big(\cap_{i\in\{1,\dots,\ell\}}\{Z_i = 1\}\big) = \prod_{i=1}^{\ell}\Pr(Z_i = 1 \mid Z_1 = 1 \cap \cdots \cap Z_{i-1} = 1)$$

Now, for any $j \leq i-1$, $Z_j = 1$ is equivalent to $X_j^{(1)} + X_j^{(2)} + X_j^{(3)} \geq 2$, so $\mathrm{E}\big(X_j \mid Z_j = 1\big) = \mathrm{E}\Big(X_j \mid X_j^{(1)} + X_j^{(2)} + X_j^{(3)} \geq 2\Big) \geq \mathrm{E}\big(X_j\big)$, where the inequality comes from the fact that the three $X_j^{(r)}$ are non-negative and that X_j is represented by one of these.

Hence, using (2), we also have

$$\Pr\big(X_i = 1 \mid Z_j = 1\big) = \frac{k - \mathrm{E}\big(X_1 \mid Z_j = 1\big) - \cdots - \mathrm{E}\big(X_{i-1} \mid Z_j = 1\big)}{n-i} \leq \frac{k - \mathrm{E}(X_1) - \cdots - \mathrm{E}(X_{i-1})}{n-i} = \Pr(X_i = 1).$$

Since (1) is monotone increasing in $\Pr(X_i = 1) \in [0,1]$, we have

$$\Pr\big(Z_i = 1 \mid Z_j = 1\big) \leq \Pr(Z_i = 1),$$

so altogether

$$\Pr\big(\cap_{i \in \{1,\dots,\ell\}} \{Z_i = 1\}\big) \leq \prod_{i=1}^{\ell} \Pr(Z_i = 1),$$

which proves the 1-negative correlation.

We can easily prove the 0-negative correlation in a symmetric fashion. Formally, if we consider $x^{(1)}, x^{(2)}, x^{(3)} \in \{0,1\}^n$ to be drawn uniformly at random and independently from the set of bit strings having $n-k$ zero-bits and redefine $X_i^{(\cdot)}$ to be the complement of the i-th bit value, then the very same proof as above applies except that any appearance of the number k of one-bits is replaced with the number $n-k$ of zero-bits. □

Finally, we define a commonly used notion of high probability.

Definition 1. We say that an event A happens *with high probability* if $\Pr(A) \geq 1 - n^{-c}$ for every constant $c > 0$.

2.4. Tools for the analysis of the cGA

In this subsection, we collect results applying to the behavior of the cGA in concrete settings, most notably when optimizing the well-known function $\textsc{OneMax}(x) = \sum_{i=1}^n x_i$. We start out with a general, simplifying assumption common in the runtime analysis of the cGA [43,10]: we assume that the frequencies $p_{t,i}$ are *well-behaved* (which term was introduced in [10]) in that they only take values of the type $1/n + i/K$ for a non-negative integer i and there is an i^* such that $1/n + i^*/K = 1 - 1/n$. Essentially, this can be achieved if $1/K$ divides $1 - 2/n$. Having well-behaved frequencies avoids some extra cases in the analyses that could appear when a frequency is only one increment or decrement away from a border, i.e., $1/n$ or $1 - 1/n$, but do not represent a crucial restriction. We therefore assume well-behaved frequencies in the rest of this work.

The progress of the cGA on OneMax and Jump functions is usually analyzed using a so-called potential function that essentially measures the distance of the frequencies from the vector $(1-1/n, 1-1/n, \dots, 1-1/n)$, i.e., the vector maximizing the probability of sampling the all-ones string. This function is called Φ_t here, following the convention from Sudholt and Witt [43]:

$$\Phi_t := n - \sum_{i=1}^n p_{t,i}. \tag{3}$$

We remark that in the analysis of the cGA on Jump [10], this potential function is denoted by D_t. Due to the borders on the frequencies, it always holds that $\Phi_t \geq n \cdot 1/n = 1$. Often, depending on which perspective is more convenient, we also study the negated potential function

$$P_t := n - \Phi_t = \sum_{i=1}^n p_{t,i}. \tag{4}$$

We say that an offspring $x \in \{0,1\}^n$ is *sampled according to frequency vector* $\boldsymbol{p}_t = (p_{t,1}, \dots, p_{t,n})$ if bit i is set to 1 with probability $p_{t,i}$ and to 0 with the remaining probability independently for all $i \in \{1, \dots, n\}$, i.e., according to the procedure of the cGA. Clearly, $P_t = \sum_{i=1}^n p_{t,i}$ then equals the expected number of one-bits of an offspring sampled with frequency vector $\boldsymbol{p}_t$. Using straightforward Chernoff bounds, we find that the actual number of one-bits is concentrated around $P_t = n - \Phi_t$. The following lemma is essentially a reformulation of Doerr [12, Lemma 5].

Lemma 7 *(cf. Lemma 5 in [12]). Let p_t be a current frequency vector, let $\Phi_t = n - \sum_{i=1}^n p_{t,i}$ and let x be sampled according to p_t. Then, for any upper bound $\Phi^+ \geq \Phi_t$ and all $\delta > 0$ it holds that*

$$\Pr\big(n - |x|_1 \geq (1+\delta)\Phi^+\big) \leq e^{-\min\{\delta, \delta^2\}\Phi^+/3}.$$

For any lower bound $\Phi^- \leq \Phi_t$ and all $\delta \in [0,1]$ it holds that

$$\Pr\big(n - |x|_1 \leq (1-\delta)\Phi^-\big) \leq e^{-\delta^2\Phi^-/2}.$$

Analogous statements hold for the random variable $|x|_1$ and deviations from its expected value P_t, i.e., $\Pr\big(|x|_1 \geq (1+\delta)P^+\big) \leq e^{-\min\{\delta,\delta^2\}P^+/3}$ as well as $\Pr\big(|x|_1 \leq (1-\delta)P^-\big) \leq e^{-\delta^2 P^-/2}$ for any upper bound P^+ and lower bound P^- on P_t.

The deviation of potential within a number k of steps will be bounded from above using the tail bound from Lemma 5. Here it is crucial to note that the scaled potential difference $K(P_{t+1} - P_t)$ equals $D_t := |\sum_{i=1}^n (x_i - y_i)|$ or its negation $-D_t$, where x and y are the two offspring of the cGA sampled at time t and the sign is determined from the ranking of the offspring. We recover the scenario of Lemma 5 by pessimistically assuming that the potential grows by $|D_t|/K$ in every step and noting that the random bitwise differences $x_i - y_i$ are distributed on $\{-1,0,1\}$ with zero expectation and drawn independently of each other (more precisely, $x_i - y_i$ is independently set to each of the two non-zero values with probability $2p_{t,i}(1-p_{t,i})$). Hence, taking care of the scaling by $1/K$, we obtain the following corollary.

Corollary 1. *Consider the potentials $P_t = \sum_{i=1}^n p_{t,i}$ and $\Phi_t = n - P_t$ of the cGA working on an arbitrary fitness function. For an arbitrary number k of steps, let $Z = |\Phi_{t+k} - \Phi_t| = |P_{t+k} - P_t|$. Then it holds for any $\lambda > 0$ that*

$$\Pr\big(Z \geq (k\sqrt{n} + \lambda)/K\big) \leq e^{-\lambda^2/(2kn\log^2 n)} + 2ke^{-(\log^2 n)/2}.$$

In addition to analyzing the dynamics of the potential Φ_t, which involves all frequencies, we shall also study the behavior of a single frequency $p_{t,i}$ for an arbitrary but fixed bit i over time. The following lemma from Sudholt and Witt [43, Lemma 3] bounds the drift of such a frequency towards the optimal value $1 - 1/n$.

Lemma 8 *(Lemma 3 in [43]). Consider the cGA on* OneMax *and an arbitrary but fixed $i \in \{1,\dots,n\}$. Let $\Delta_t = p_{t+1,i} - p_{t,i}$. If $1/n + 1/K \leq p_{t,i} \leq 1 - 1/n - 1/K$ then*

$$\mathrm{E}\big(\Delta_t \mid p_{t,i}\big) \geq \frac{2}{11}\frac{p_{t,i}(1-p_{t,i})}{K}\left(\sum_{j\neq i} p_{t,j}(1-p_{t,j})\right)^{-1/2}.$$

We will also need certain ingredients of the proof of Lemma 8. According to the presentation in Sudholt and Witt [43], the one-step change of a frequency in the context of the cGA optimizing OneMax is given by a so-called *superposition* of so-called biased and random-walk steps: For $\Delta_t := p_{t+1,i} - p_{t,i}$ it holds that

$$\Delta_t = F_t \cdot \mathbb{1}_{R_t} + B_t \cdot \mathbb{1}_{\overline{R_t}}, \tag{5}$$

where F_t is the random change in the case of a *random-walk (rw)* step, i.e., when the outcome of bit i cannot influence the ranking of the two offspring of the cGA and B_t the change in the case of a *biased (b) step* when the ranking can change depending on the outcome; moreover, R_t is the event of a rw-step happening, more precisely the event that the number of one-bits in the $n-1$ positions other than i of the two offspring differs by an absolute value of at least 2. Arguing that $p_{t,i}$ moves only if the two offspring have different bit values, implying the distributions

$$F_t = \begin{cases} -\frac{1}{K} & \text{with probability } p_{t,i}(1-p_{t,i}), \\ +\frac{1}{K} & \text{with probability } p_{t,i}(1-p_{t,i}), \\ 0 & \text{with the remaining probability,} \end{cases}$$

and

$$B_t = \begin{cases} \frac{1}{K} & \text{with probability } 2p_{t,i}(1-p_{t,i}), \\ 0 & \text{with the remaining probability,} \end{cases}$$

and analyzing $\Pr(R_t)$ depending on the current Φ_t-value, the result of Lemma 8 is obtained from (5) via the linearity of expectation.

Finally, we shall need the following result to bound genetic drift. With respect to the cGA, similar statements were presented in Sudholt and Witt [43] and Witt [45]. The present formulation uses notation and arguments from Doerr and Zheng [14], who give the to date most comprehensive treatment of genetic drift.

Lemma 9 *(cf. [14]). Let $p_{t,i}$, $t \geq 0$, denote the stochastic process described by an arbitrary but fixed frequency i that performs either a b-step increasing the frequency by $1/K$ with probability $2q_{t,i} := 2p_{t,i}(1-p_{t,i})$ and staying unchanged with probability $1 - 2q_{t,i}$, or a rw-step that increases by $1/K$, decreases by $1/K$ and stays unchanged with probability $q_{t,i}$, $q_{t,i}$ and $1 - 2q_{t,i}$, respectively.*

Then for all $\gamma > 0$ and $T \in \mathbb{N}$, the following two inequalities hold:

(a) *the probability that there is a point in time* $t \in \{0, \dots, T\}$ *such that* $p_{t,i} < 1/2 - \gamma$ *is bounded from above by* $\exp(-\gamma^2 K^2/(2T))$,
(b) *using the initial frequency* $p_{0,i} = a < 1/2$ *instead of* $1/2$, *the probability that there is a point in time* $t \in \{0, \dots, T\}$ *such that* $p_{t,i} < a - \gamma$ *is bounded from above by* $\exp(-\gamma^2 K^2/(2T))$.

Proof. The first part was already shown in Doerr and Zheng [14, Th. 4.2]. The second part follows by re-doing the underlying analysis for the initial frequency value a. □

3. Jump with offset fitness gap

We now investigate how efficient the algorithms introduced in Section 2 are at crossing the shifted fitness gap of the JumpOffset function. Not surprisingly, the two crossover-based algorithms do not have any difficulties in crossing the gap. However, this is not equivalent to finding the optimum (the all-ones string) efficiently.

Theorem 4. *Consider the* JumpOffset *function for arbitrary* $m < n/4$. *For any constant* $c > 0$, *the voting algorithm (Algorithm 2) with* $\mu \geq 32(c+1)n \log n$ *samples the global optimum of* JumpOffset *in time* $O(\mu)$ *with probability at least* $1 - 1/n^c - 2^{-\Omega(n)}$.

With probability $1 - 2^{-\Omega(n)}$, *the hybrid GA (Algorithm 1) on* JumpOffset *will output a search point* x *such that* $|x|_1 \leq (27/32 + \epsilon)n$ *where* $\epsilon > 0$ *is an arbitrary constant. In particular, it will not find the optimum in this case; moreover, the search point* x *will be in the gap of* JumpOffset *if* $m \geq (3/32 + \epsilon)n$.

Proof. The first statement is a direct consequence of Corollary 3.3 in Rowe and Aishwaryaprajna [41], stating that the voting algorithm on the classical Jump for $m < (1-a)n/2$, where $a > 0$ is an arbitrary constant, will only sample search points outside the gap with overwhelming probability $1 - 2^{-\Omega(n)}$ and then behave as if it was optimizing OneMax. Since Corollary 3.3 in Rowe and Aishwaryaprajna [41] gives the same bound on the success probability as the OneMax analysis (Theorem 3.2 in the paper), it seems to us that the failure probability $2^{-\Omega(n)}$ related to sampling in the gap is not treated explicitly in the corollary, and we have included it here to be on the safe side.

The second statement requires an analysis of the majority-vote crossover, where we build upon the improved analysis of the hybrid GA given in Rowe and Aishwaryaprajna [41]. With probability $1 - 2^{-\Omega(n)}$, the hybrid GA will initialize all individuals with less than $n/2 + \epsilon n$ one-bits and then create three individuals uniformly distributed at the start of the gap, i.e., having $3n/4$ ones-bits. The location of these bits is independent of each other among these individuals. Letting $k = 3n/4$, the probability that a bit is sampled as one in the majority-vote crossover is

$$3\frac{n-k}{n}\left(\frac{k}{n}\right)^2 + \left(\frac{k}{n}\right)^3 = \frac{27}{32}.$$

The random variables Z_i denoting the outcome of majority-vote crossover for bit i are negatively correlated according to Lemma 6. Using Chernoff bounds for negatively correlated (non-independent) random variables (Theorem 1.10.24 in Doerr 11), the probability of having more than $(27/32 + \epsilon)n$ positions set to 1 in majority-vote crossover is $2^{-\Omega(n)}$. □

The second statement of Theorem 4 deserves some discussion. Due to the lack of a concise algorithm description in Whitley et al. [44], it is not very clear whether the hybrid GA from that paper would continue improving the offspring created by majority-vote crossover with next-ascent hillclimbing or whether it would stop after the first crossover, which was sufficient for the analysis on the original Jump function. If hillclimbing is continued after crossover, then according to Theorem 4, it would nevertheless again move to the start of the gap at $3n/4$ with overwhelming probability if $m \geq (3/32 + \epsilon)n$. However, this is a very large gap size. Smaller gap sizes would allow the crossover offspring to be "behind the gap", i.e., to have more than $3n/4 + m$ one-bits with high probability, from where the optimal all-ones string could be reached efficiently using hillclimbing.

We now turn to the cGA (always assuming well-behaved frequencies as defined in Section 2.4), which can efficiently overcome gap sizes up to $c \ln n$ for a small constant $c > 0$ on the original (non-shifted) Jump function. For the JumpOffset function, we obtain that gaps of size almost $\sqrt{n}$ can be overcome. In the following theorem, we derive a result that holds with high probability (Definition 1) and include a helper result on the potential Φ_t (see (3)) that will turn out useful later.

Theorem 5. *Let* $K \geq \lceil c\sqrt{n} \log n \rceil$ *for a sufficiently large constant* $c > 0$ *and* $K = \text{poly}(n)$. *Then the cGA, under the well-behaved frequencies assumption, with parameter* K *on the fitness function* JumpOffset_m *with* $m = O(\sqrt{n}/\log^5 n)$ *creates a frequency vector having potential* $\Phi_t \leq 9216$ *and samples the global optimum in* $O(K\sqrt{n})$ *steps with high probability. The last bound is* $O(n \log n)$ *for* $K = \lceil c\sqrt{n} \log n \rceil$.

The proof of this theorem is long and in parts rather technical. It builds upon the analysis of the cGA on OneMax by Sudholt and Witt [43] rather than the analysis on Jump by Doerr [12] since the former is based on a study of the stochastic behavior of individual frequencies that is useful when applying concentration results like the negative drift theorem and variable drift analysis with tail bounds.

Roughly, the analysis is divided into different claims according to the state of the potential $P_t = n - \Phi_t = \sum_{i=1}^{n} p_{t,i}$. The overall idea is that the potential P_t will reach a state close to n that allows efficient sampling of the all-ones string. This part of the analysis builds upon the insight that the cGA behaves as on OneMax if the samples do not fall into the gap region of JumpOffset. This allows us to re-use major parts of the machinery from Sudholt and Witt [43, Th. 2 and 5] who give a tail bound on the optimization time of the cGA on OneMax using a drift analysis with tail bounds. The underlying drift is not affected significantly when the current state of the potential makes sampling in the gap unlikely.

The crucial difference appears in situations when P_t is in a so-called critical region ($P_t \geq 3n/4 - \sqrt{n}\ln^2 n$ and $P_t < 3n/4 + m + \sqrt{n}\ln^2 n$) where the probability of sampling offspring in the gap of JumpOffset can be non-negligible. Using additional drift arguments, we will bound the time for the potential to pass the critical region and then show that the probability of returning to the critical region afterwards is small. Since the critical region is small, we obtain that the drift on the OneMax-part of the function still dominates the overall drift of the algorithm. Moreover, since the critical region is left quickly, we can control the accumulated effect of genetic drift. In the following, we will use small headings in italics to structure the proof according to the above-mentioned states.

Proof of Theorem 5. Overall, we aim at following the analysis of the cGA on OneMax from Sudholt and Witt [43, Th. 2 and 5] since the fitness function equals OneMax with a constant additive term except for search points whose number of ones is in the so-called gap region $G := (3n/4, 3n/4 + m)$. The key arguments of the OneMax-analysis in Sudholt and Witt [43] break down into a stochastic drift analysis of single frequencies (based on the superposition equality (5)) along with an analysis of genetic drift, i.e., the situation that frequencies approach the lower border due to stochastic fluctuations. We will show that steps that sample offspring in the gap change the effects of stochastic and genetic drift at most by lower-order terms. Then, in a fixed period of $\Theta(K\sqrt{n})$ steps, sampling the optimum will turn out to occur with high probability. In the following, we assume that all frequencies are bounded from below by at least $1/4$ before the optimum is reached and bound the failure probability at the end of the proof.

We consider an arbitrary fixed bit i and its frequency $p_{t,i}$, where $p_{t,i} \leq 1 - 1/n - 1/K$, over time and extend the superposition equality (5) with respect to the event G_t that the number of ones I at positions other than i in at least one offspring is in the interval $[3n/4 - 1, 3n/4 + m + 1]$ and that the other offspring has $I - 1$, I, or $I + 1$ ones at those positions. We call a step where G_t occurs a *g-step* and note that only such steps can have a negative drift of the frequency $p_{t,i}$ since event G_t is necessary for at least one offspring to be in the gap and for the outcome of bit i to be relevant for the fitness ranking of the two offspring.

In a g-step, we pessimistically assume that the frequency decreases by $1/K$ if it can change. Such a change is possible if bit i is sampled differently in the two offspring. This leads to the inequality:

$$\Delta_t \geq F_t \cdot \mathbb{1}_{R_t} + B_t \cdot \mathbb{1}_{B_t} - \frac{2p_{t,i}(1-p_{t,i})}{K} \cdot \mathbb{1}_{G_t},$$

where we have $B_t = \overline{R_t} \cap \overline{G_t}$. With this bound on Δ_t at hand, we distinguish between two cases with respect to the potential $P_t = \sum_{i=1}^{n} p_{t,i}$ according to the so-called critical region $[3n/4 - \sqrt{n}\ln^2 n, 3n/4 + m + \sqrt{n}\ln^2 n]$.

The potential is outside the critical region If $P_t < 3n/4 - \sqrt{n}\ln^2 n$ or $P_t > 3n/4 + m + \sqrt{n}\ln^2 n$, then sampling in the gap has superpolynomially small probability $2^{-\Omega(\ln^2 n)}$ according to Lemma 7. We pessimistically assume that all frequencies are decreased in such a case. This changes the drift of the potential function by at most $(n/K)2^{-\Omega(\ln^2 n)} = 2^{-\Omega(\ln^2 n)}$. Hence, the polynomial bound $\Omega(\sqrt{\Phi_t})/K$ on the drift of $\Phi_t = n - P_t$ derived for OneMax in Sudholt and Witt [43] (presented and adapted in the following analysis of the critical region) changes only by a superpolynomially small term in this case and will change the final result only by a lower-order term.

The potential is inside the critical region; analysis of drift We now turn to the other case, where P_t is in the interval $[3n/4 - \sqrt{n}\ln^2 n, 3n/4 + m + \sqrt{n}\ln^2 n]$. A key observation is that we now have $\sigma_t^2 := \sum_{i=1}^{n} p_{t,i}(1 - p_{t,i}) \geq \Phi_t/4 = (n - P_t)/4 = \Omega(n)$ since all $p_{t,i} \geq 1/4$. For each sampled offspring, the number of one-bits follows a Poisson-binomial distribution with parameters $p_{t,1}, \ldots, p_{t,n}$, and by Lemma 2, the probability of sampling a search point with exactly j one-bits, for any $0 \leq j \leq n$, is at most $1/(2\sigma_t) = O(1/\sqrt{n})$. This has two useful implications:

(a) The probability of an offspring sampled in the gap is at most $O(m/\sqrt{n}) = O(1/\log^5 n)$ by a simple union bound.
(b) We have $\Pr(G_t) = O(1/(\sqrt{n}\log^5 n))$ since for G_t to happen, one search point has to be in the gap (probability $O(1/\log^5 n)$) and the number of one-bits at positions $j \neq i$ in the two search points must differ by no more than an absolute value of 1 (probability $O(1/\sqrt{n})$).

Let D_t be the event that the two offspring have the same number of ones at the positions $j \neq i$. If D_t happens, then the frequency $p_{t,j}$ does not perform a random step but either a b-step (leading to positive drift) or a g-step (pessimistically leading to negative drift). According to Lemma 3 in Sudholt and Witt [43], we have

$$\Pr(D_t) \geq \frac{1}{11}\left(\sum_{j\neq i} p_{t,j}(1-p_{t,j})\right)^{-1/2}.$$

In the following, we estimate $\Pr(B_t) \geq \Pr(D_t) - \Pr(G_t)$. Combining this with Properties (a) and (b) and plugging in the expected values $\mathrm{E}(F_t \mid p_{t,i}) = 0$ and $\mathrm{E}(B_t \mid p_{t,i}) = 2p_{t,i}(1-p_{t,i})/K$ derived in Section 2.4, we have for the drift of $p_{t,i}$ that

$$\mathrm{E}(\Delta_t \mid p_{t,i}) \geq \frac{2}{11}\frac{p_{t,i}(1-p_{t,i})}{K}\left(\sum_{j\neq i} p_{t,j}(1-p_{t,j})\right)^{-1/2} - O\left(p_{t,i}(1-p_{t,i})/(K\sqrt{n}\log^5 n)\right),$$

which, since $\left(\sum_{j\neq i} p_{t,j}(1-p_{t,j})\right)^{-1/2} \geq \left(\sum_{j=1}^{n} p_{t,j}(1-p_{t,j})\right)^{-1/2} \geq 1/\sqrt{n}$ implies that the negative term is of lower order. Hence,

$$\mathrm{E}(\Delta_t \mid p_{t,i}) \geq \frac{1}{6}\frac{p_{t,i}(1-p_{t,i})}{K}\left(\sum_{j\neq i} p_{t,j}(1-p_{t,j})\right)^{-1/2} \geq \frac{1}{6}\frac{p_{t,i}(1-p_{t,i})}{K\sigma_t}$$

for n large enough, which is asymptotically the same drift as in the pure OneMax-case (see Lemma 8 for comparison).

For the drift of $\Phi_t = \sum_{i=1}^{n}(1-p_{t,i})$ this implies, if all $p_{t,i} \leq 1 - 1/n - 1/K$, that

$$\begin{aligned}\mathrm{E}(\Phi_t - \Phi_{t+1} \mid p_{t,1},\dots,p_{t,n}) &= \sum_{i=1}^{n}\mathrm{E}(\Delta_t \mid p_{t,i}) \geq \frac{1}{6}\frac{\sum_{i=1}^{n} p_{t,i}(1-p_{t,i})}{K\sigma_t} \\ &= \frac{1}{6}\frac{\sigma_t}{K} = \frac{1}{6}\frac{\sqrt{\sum_{i=1}^{n} p_{t,i}(1-p_{t,i})}}{K} \geq \frac{1}{6}\frac{\sqrt{\sum_{i=1}^{n}(1-p_{t,i})/4}}{K} \geq \frac{1}{12}\frac{\sqrt{\Phi_t}}{K},\end{aligned}$$

where we have used that $p_{t,i} \geq 1/4$. We remark that similar arguments, combining the individual drifts of frequencies into a drift of the potential using linearity of expectation, have been used extensively in Sudholt and Witt [43] and Lengler et al. [32].

The frequencies at the upper border $1-1/n$ can change with probability at most $2(1/n)(1-1/n)$ at most since the two offspring must be sampled differently at the corresponding bit. Pessimistically changing any increase possible if the frequency was not at a border to a decrease in such a case, the previous assumption of no frequencies at the upper border overestimates the drift of Φ_t by at most $2n(1/n)(1-1/n)(2/K) \leq 4/K$. We get the unconditional bound

$$\mathrm{E}(\Phi_t - \Phi_{t+1} \mid p_{t,1},\dots,p_{t,n}) \geq \frac{1}{12}\frac{\sqrt{\Phi_t}}{K} - \frac{4}{K},$$

which, for $\Phi_t \geq 96^2 = 9216$, results in

$$\mathrm{E}(\Phi_t - \Phi_{t+1} \mid p_{t,1},\dots,p_{t,n}) = \mathrm{E}(P_{t+1} - P_t \mid p_{t,1},\dots,p_{t,n}) \geq \frac{\sqrt{\Phi_t}}{24K}, \tag{6}$$

where we, for future use, have noted that the drift of P_t is the negated drift of Φ_t. We recall from the analysis of P_t outside the critical region that (6) holds for all $\Phi_t \geq 9216$ and not only when P_t is in the critical region.

The potential leaves the critical region quickly and is unlikely to return to it We now analyze the time until P_t has clearly left the critical region, including some "safety margin", and reached a value of at least $b := 3n/4 + m + 2\sqrt{n}\ln^2 n$, i.e., by $\sqrt{n}\ln^2 n$ larger than the end of the critical region. This piece of analysis will be crucial to bound the overall effect of genetic drift, analyzed in the final part of this proof.

To bound the time for P_t to leave the critical region towards b, we shall use additive drift analysis (Th. 2.3.1 in [31]; alternatively Th. 2 in the special case that $h(x)$ is uniformly bounded from below by some δ for all $x \geq x_{\min}$). Since P_t is not necessarily monotonically increasing during the run, we similarly as before introduce a safety margin and consider the point $s := 3n/4 - 2\sqrt{n}\ln^2 n$, i.e., by $\sqrt{n}\ln^2 n$ smaller than the start of the critical region. We will prove below that the probability of P_t dropping below s after having reached the start $3n/4 - \sqrt{n}\ln^2 n$ of the critical region is superpolynomially small and assume now that this does not happen within any polynomial number of steps. By a union bound, the failure probability is still polynomially small and will be included in the failure probability stated in the theorem.

Recall that the drift of P_t is $\Omega(\sqrt{n}/K)$ in the critical region (6) and even in the enlarged region $[s,b]$. Let T be the first point in time where $P_t \geq b$. To apply the additive drift theorem, we have to bound P_T from above to handle the fact that the target b can be overshot. Applying Corollary 1 for $k=1$ and $\lambda = \sqrt{n}\log^2 n$ (a crude estimate sufficient for our purposes), we have for any $t \geq 0$ that $P_{t+1} - P_t > (\sqrt{n}\ln^2 n)/K$ with probability $e^{-\Omega(\log^2 n)}$. By a union bound, this also applies throughout any phase of polynomial length. Hence, for any polynomial value of T, we have $P_T \leq b + (\sqrt{n}\ln^2 n)/K \leq b + \sqrt{n}\ln^2 n$

with probability $1 - e^{-\Omega(\log^2 n)}$. We assume this to happen and subsume the failure case in the failure probability of this theorem. Altogether, the length of interval to cross is at most $b + \sqrt{n}\ln^2 n - s = 5\sqrt{n}\ln^2 n + m = O(\sqrt{n}\ln^2 n)$. Hence, using the additive drift theorem, the expected time from entering the critical region until leaving it and reaching at least b is $O(K(\sqrt{n}\ln^2 n)/\sqrt{n}) = O(K\ln^2 n)$. Denoting by C the implicit constant in the last O-term and applying Markov's inequality, the time is bounded by $2CK\ln^2 n$ with probability at least $1/2$. If this time is not sufficient, the same argumentation holds independently of earlier phases with the current value of $P_t \in [s, b)$ as starting value. (Recall that we assume that P_t will not drop below s after having reached the start $3n/4 - \sqrt{n}\ln^2 n$ of the critical region in any polynomial number of steps.) Hence, repeating this argumentation with $\ln^2 n$ phases of length $2CK\ln^2 n$, the time to reach b after entering the critical region is at most $O(K\log^4 n)$ with probability $1 - n^{-\Omega(\log n)}$.

After P_t reaching $b = 3n/4 + m + 2\sqrt{n}\ln^2 n$ for the first time, we show that it is superpolynomially unlikely to drop to $a := 3n/4 + m + \sqrt{n}\ln^2 n$, i.e., the end of the critical region, again. The same analysis proves it superpolynomially unlikely to drop below $s = 3n/4 - 2\sqrt{n}\ln^2 n$ again after reaching the start $3n/4 - \sqrt{n}\ln^2 n$ of the critical region, so that we present the arguments only with respect to the interval $[a, b]$ in the following. In our analysis, we apply the negative drift theorem with scaling (Theorem 1) on the P_t-process (recall that $P_t = n - \Phi_t$). The first condition of that theorem, the bound on the drift, given a state in $[a, b]$, has already been established to be $\Omega(\sqrt{n}/K) = \Omega(\sqrt{P_t}/K)$ above in (6). For convenience, we from now on consider the drift of KP_t and have $\epsilon = c'\sqrt{P_t}$ in the notation of the drift theorem, for an appropriate constant $c' > 0$. We now turn to the second condition of the drift theorem. For an appropriately defined function r, we have to show

$$\Pr(K|P_{t+1} - P_t| \geq jr;\ P_t > a \mid P_t) \leq e^{-j}.$$

In the following, we will omit the conditioning on $P_t > a$ and P_t for readability.

We observe that, depending on which offspring is ranked fittest, either $K(P_{t+1} - P_t)$ or $K(P_t - P_{t+1})$ is bounded from above by the sum X of n independent random variables $X_1, \dots, X_n$ with support $\{-1, 0, 1\}$ such that

$$X_i = \begin{cases} -1 & \text{with probability } p_{t,i}(1 - p_{t,i}) \\ 1 & \text{with probability } p_{t,i}(1 - p_{t,i}) \\ 0 & \text{with probability } 1 - 2p_{t,i}(1 - p_{t,i}) \end{cases}$$

(this argument already appeared in Section 2.4 to prove Corollary 1). We again use the notation $\sigma_t^2 = \sum_{i=1}^n p_{t,i}(1 - p_{t,i}) = \mathrm{Var}(X)$ and note that $\sigma_t^2 = \sum_{i=1}^n p_{t,i}(1 - p_{t,i}) \geq n(1/n)(1 - 1/n) \geq 1/2$ due to the borders on the frequencies. Applying Lemma 1 with these parameters and bounds as well as $\mathrm{E}(X) = 0$ and $b = 1$, we conclude for all $j \geq 1$ that

$$\Pr(X > 12j\sigma_t) \leq \exp(-(1/3)\min\{144j^2, 12j\sigma_t\}) \leq \exp(-\min\{144j^2/3, 2j\}) \leq e^{-2j}$$

and, symmetrically,

$$\Pr(X < -12j\sigma_t) \leq e^{-2j}$$

so that

$$\Pr(|X| > 12j\sigma_t) \leq 2e^{-2j} \leq e^{-j}$$

since $j \geq 1$. Combining with $\sigma_t \leq \sqrt{P_t}$, this immediately proves

$$\Pr\left(K|P_{t+1} - P_t| > 12j\sqrt{P_t}\right) \leq e^{-j}.$$

The last inequality satisfies the second condition of Theorem 1 with $r = 12\sqrt{P_t}$.

The third condition holds for the following reason. On the one hand, $r = 12\sqrt{P_t} \geq 1$ since $P_t = \Omega(n)$. On the other hand, recalling $\ell = b - a = \sqrt{n}\ln^2 n$ and using the trivial estimations $P_t \leq n$ and $\epsilon = \Omega(\sqrt{n}) \geq 1$, we have

$$r^2 \leq 144P_t \leq \frac{c''\sqrt{P_t}\sqrt{n}\ln^2 n}{132\log n} \leq \frac{\epsilon\sqrt{n}\ln^2 n}{132\log(n/\epsilon)} \leq \frac{\epsilon\ell}{132\log(r/\epsilon)},$$

where we assumed that n is large enough, used $\ell = \Omega(\sqrt{n}\ln^2 n)$ and chose the constant $c'' > 0$ small enough.

Now, Theorem 1 yields for the first hitting time T^* of $P_t \leq a$, conditioned on $P_0 \geq b$, that

$$\Pr\left(T^* \leq e^{\epsilon\ell/(132r^2)}\right) = O(e^{-\epsilon\ell/(132r^2)}).$$

Since

$$\frac{\epsilon}{132r^2} = \frac{\Omega(\sqrt{P_t})}{132(12\sqrt{P_t})^2} = \Omega(1/\sqrt{n})$$

and $\ell = b - a = \Omega(\sqrt{n}\log^2 n)$, we have for a sufficiently small constant $\kappa > 0$ that

$$\Pr\left(T^* \leq n^{\kappa \log n}\right) = n^{-\Omega(\log n)}.$$

Altogether, the probability that P_t drops again below the value $3n/4 + m + \sqrt{n}\ln^2 n$, the end of the critical region, within polynomial time after having reached $b = 3n/4 + m + 2\sqrt{n}\ln^2 n$ is superpolynomially unlikely. We remark again that the same analysis holds for the interval $[3n/4 - 2\sqrt{n}\ln^2 n, 3n/4 - \sqrt{n}\ln^2 n]$, considered above when we analyzed the time to leave the critical region towards b.

Bounding the time and success probability for sampling the optimum We finally use variable drift analysis with tail bounds like in the proof of Theorem 5 in Sudholt and Witt [43] to analyze the time T until the potential Φ_t is reduced to at most 9216. Although not strictly needed in all its aspects, we briefly bound the expected value of T to illustrate a main idea of variable drift analysis. Let $h(x) = \sqrt{x}/(24K)$ and $x_{\min} = 9216$. Using Theorem 2, we have for the expected time (conditional on none of the failure events mentioned above happening)

$$\mathrm{E}(T \mid \Phi_0) \leq \frac{24K \cdot 9216}{\sqrt{9216}} + \int_{9216}^{n} \frac{24K}{\sqrt{x}}\,\mathrm{d}x = O(K\sqrt{n}). \tag{7}$$

(Note that we actually analyze the time for the potential to reach zero, pessimistically identifying all states of potential less than 9216 with 0 to handle the event that the exact value of 9216 may have been undershot at time T).

However, we have to show that $T = O(K\sqrt{n})$ also holds with high probability. To this end, we will bound the moment-generating function (mgf.) of the change of the above integral in the same way as in the proof of Theorem 5 in Sudholt and Witt [43], literally copying major parts of the analysis of this mgf., and then invoke Theorem 3 to show a tail bound on the time to sample the all-ones string. More precisely, we will bound the mgf. of (a stochastic upper bound on) the absolute value of

$$\int_{\Phi_{t+1}}^{\Phi_t} \frac{1}{h(\max\{x, x_{\min}\})}\,\mathrm{d}x \leq \int_{\Phi_{t+1}}^{\Phi_t} \frac{K'}{\sqrt{x}}\,\mathrm{d}x,$$

where we use $K' = 24K$ to improve readability.

We first prove that $\Phi_{t+1} \geq \Phi_t/4$ with overwhelming probability. To this end, let M_t be the number of frequencies at time t that are less than $1 - 1/n$, i.e., can increase in the step to time $t+1$. Note that each such increment is $1/K$ and corresponds to decreasing Φ_t by $1/K$. We now distinguish between two cases. If $M_t \leq K$, we pessimistically assume all M_t frequencies to increase and estimate $\Phi_{t+1} \geq \Phi_t - M_t/K \geq \Phi_t - 1 \geq \Phi_t/4$, where the last inequality used $\Phi_t \geq x_{\min} = 9216$. If $M_t > K$, we use that each frequency $p_{t,i}$ increases with probability at most $2p_{t,i}(1-p_{t,i}) \leq 1/2$. Hence, by Chernoff bounds, the number of increasing frequencies is at most $3M_t/4$ with probability at least $1 - 2^{-\Omega(M_t)} = 1 - 2^{-\Omega(K)} = 1 - 2^{-\Omega(\sqrt{n}\log n)}$ by our assumption on K, so $\Phi_{t+1} \geq \Phi_t - \frac{3M_t}{4K}$ with this probability. On the other hand, $\Phi_t \geq M_t/K$ since by the well-behaved frequencies assumption, each of the M_t frequencies is at most $1 - 1/n - 1/K$. So, altogether, with probability $1 - 2^{-\Omega(\sqrt{n}\log n)}$ it holds that $\Phi_{t+1} \geq \Phi_t/4$. In the following, we assume this to hold throughout a period of $c'''K\sqrt{n}$ steps for an arbitrary constant $c''' > 0$. Since $K = \mathrm{poly}(n)$, the corresponding probability is still $1 - 2^{-\Omega(\sqrt{n}\log n)}$ by a union bound and will be subsumed by the failure probability stated in the theorem.

Using our assumption $\Phi_{t+1} \geq \Phi_t/4$, we obtain

$$\left|\int_{\Phi_{t+1}}^{\Phi_t} \frac{1}{h(\max\{x, x_{\min}\})}\,\mathrm{d}x\right| \prec \left|\int_{\Phi_{t+1}}^{\Phi_t} \frac{K'}{\sqrt{x}}\,\mathrm{d}x\right| \prec \frac{K'|\Phi_{t+1} - \Phi_t|}{\sqrt{\Phi_t/4}}.$$

We are left with an analysis of $|\Phi_{t+1} - \Phi_t|$. We note that for any bit i, its frequency changes by an absolute value of at most $1/K$ with probability $2p_{t,i}(1-p_{t,i}) \leq 2q_{t,i}$, where $q_{t,i} = 1 - p_{t,i}$. Hence, $K|\Phi_{t+1} - \Phi_t|$ is stochastically dominated by a Poisson-binomial distribution with parameters n and $\min\{1, 2q_{t,i}\}$, where $1 \leq i \leq n$. Let A be the random variable describing this Poisson-binomial distribution. While we do not know the individual success probabilities behind A, we can relate their average $p^* := \frac{1}{n}\sum_{i=1}^{n} \min\{1, 2q_{t,i}\}$ to the potential and obtain $\Phi_t/n \leq p^* \leq 2\Phi_t/n$ since $\min\{1, 2q_{t,i}\} \geq q_{t,i}$. We next bound A by a random variable B distributed according to $B \sim np^* + \mathrm{Bin}(n, p^*) + 2$, where $\mathrm{Bin}(a, b)$ denotes a binomially distributed random variable with parameters a and b. To show this, we note that $\Pr(B \geq t \mid \boldsymbol{p}_t) \geq \Pr(A \geq t \mid \boldsymbol{p}_t)$ is trivial for $t \leq np^* + 2$ (as $\Pr(B \geq t \mid \boldsymbol{p}_t) = 1$). For $t > np^* + 2$, even the dominance $\Pr(\mathrm{Bin}(n, p^*) \geq t \mid \boldsymbol{p}_t) \geq \Pr(A \geq t \mid \boldsymbol{p}_t)$ holds by the results of Gleser [22], see Marshall et al. [34, p. 495] for a summary. Hence, recalling that we have derived a stochastic bound on $K|\Phi_{t+1} - \Phi_t|$,

$$\left| \int_{\Phi_{t+1}}^{\Phi_t} \frac{1}{h(\max\{x, x_{\min}\})} \, \mathrm{d}x \right| \prec \frac{1}{K} \frac{K'(np^* + 2 + \mathrm{Bin}(n, p^*))}{\sqrt{\Phi_t/4}} = \frac{c_1(np^* + 2 + \mathrm{Bin}(n, p^*))}{\sqrt{\Phi_t}} =: Z$$

for the constant $c_1 = 48$. (Note that the constants c_1 and c_2 that will appear in this proof are not related to the constants of the same name from Lemma 3.)

We now bound the mgf. of Z. By definition, we have

$$\mathrm{E}\left(e^{\lambda Z} \mid \boldsymbol{p}_t\right) = \mathrm{E}\left(\left(e^{\lambda(np^*+2+\mathrm{Bin}(n,p^*))}\right)^{c_1/\sqrt{\Phi_t}} \mid \boldsymbol{p}_t\right).$$

Since $\Phi_t \geq x_{\min} = 9216$, we have $c_1/\sqrt{\Phi_t} \leq 1/2 < 1$, so the function $x^{c_1/\sqrt{\Phi_t}}$ is concave. Applying Jensen's inequality gives

$$\mathrm{E}\left(e^{\lambda Z} \mid \boldsymbol{p}_t\right) \leq \mathrm{E}\left(\left(e^{\lambda(np^*+2+\mathrm{Bin}(n,p^*))}\right) \mid \boldsymbol{p}_t\right)^{c_1/\sqrt{\Phi_t}}.$$

Looking up the mgf. of a binomial distribution, we obtain

$$\mathrm{E}\left(e^{\lambda Z} \mid \boldsymbol{p}_t\right) \leq \left(e^{\lambda(np^*+2)} \cdot \left(1 - p^* + p^* e^{\lambda}\right)^n\right)^{c_1/\sqrt{\Phi_t}} = \left(e^{\lambda(p^*+2/n)} \left(1 - p^* + p^* e^{\lambda}\right)\right)^{nc_1/\sqrt{\Phi_t}}.$$

Assuming $\lambda \leq \min\{1, 1/(8c_1\sqrt{\Phi_t})\}$ and using $e^{\lambda} \leq 1 + \lambda + \lambda^2 \leq 1 + 2\lambda$, we bound the last expression from above by

$$\left(e^{\lambda(p^*+2/n)}(1 - p^* + p^*(1 + 2\lambda))\right)^{nc_1/\sqrt{\Phi_t}} = \left(e^{\lambda(p^*+2/n)}(1 + 2p^*\lambda))\right)^{nc_1/\sqrt{\Phi_t}},$$

which, since $1 + x \leq e^x$ for $x \in \mathbb{R}$, is at most

$$e^{\lambda(np^*+2)c_1/\sqrt{\Phi_t}} \cdot e^{\lambda 2p^* nc_1/\sqrt{\Phi_t}} \leq e^{\lambda 4np^* c_1/\sqrt{\Phi_t}} \leq e^{8\lambda c_1\sqrt{\Phi_t}},$$

since $p^* \geq \Phi_t/n$ and $np^* \geq 9216 \geq 2$ by our assumption that $\Phi_t \geq x_{\min} = 9216$.

Using (again) $e^x \leq 1 + 2x$ for $x \leq 1$ and recalling that $\lambda \leq 1/(8c_1\sqrt{\Phi_t})$, we arrive at the bound

$$\mathrm{E}\left(e^{\lambda Z} \mid \boldsymbol{p}_t\right) \leq 1 + 16c_1\lambda\sqrt{\Phi_t} \leq 1 + c_2\lambda\sqrt{\Phi_t} \leq 1 + c_2\lambda\sqrt{n} =: D$$

using $\Phi_t \leq n$ and the constant $c_2 = 16c_1 = 768$. Summing up, we have bounded

$$\left| \int_{\Phi_{t+1}}^{\Phi_t} \frac{1}{h(\max\{x, x_{\min}\})} \, \mathrm{d}x \right| \preceq Z$$

for a random variable Z satisfying $\mathrm{E}\left(e^{\lambda Z} \mid \boldsymbol{p}_t\right) \leq 1 + c_2\lambda\sqrt{n}$.

We now apply Theorem 3 using this bound and obtain for any $\delta > 0$ and $\eta \leq \min\{\lambda, \delta\lambda^2/(D - 1 - \lambda)\}$ that

$$\Pr(T > t) \leq e^{\eta(x_{\min}/h(x_{\min}) + \int_{x_{\min}}^{\Phi_0} 1/h(x)\,\mathrm{d}x - (1-\delta)t)}. \tag{8}$$

Plugging in D, we satisfy

$$\frac{\delta\lambda^2}{D - 1 - \lambda} = \frac{\delta\lambda^2}{(c_2\sqrt{n} - 1)\lambda} \geq \frac{\delta\lambda}{c_2\sqrt{n}}$$

since $c_2\sqrt{n} - 1 \geq 0$. Similarly, we show that

$$\frac{\delta\lambda^2}{D - 1 - \lambda} = \frac{\delta\lambda}{c_2\sqrt{n} - 1} = \frac{\delta\lambda}{768\sqrt{n} - 1} \leq \lambda$$

for all $n \geq 1$ and $\delta := 1/2$, so that only the second argument of $\min\{\lambda, \delta\lambda^2/(D - 1 - \lambda)\}$ needs to be considered. We choose $\lambda := 1/(8c_1\sqrt{n})$ and $\eta := \delta\lambda/(c_2\sqrt{n}) = c_3/n$ for the constant $c_3 = 1/(8c_1c_2)$ to satisfy the requirements on λ and η. Substituting η and δ in (8) gives the concrete tail bound

$$\Pr(T > t \mid \Phi_0) \leq e^{\Omega((x_{\min}/h(x_{\min}) + \int_{x_{\min}}^{\Phi_0} 1/h(x)\,\mathrm{d}x - t/2)/n)}.$$

Since $x_{\min}/h(x_{\min}) + \int_{x_{\min}}^{\Phi_0} 1/h(x)\,\mathrm{d}x = O(K\sqrt{n})$ (see (7)), we can choose $t = c_4 K\sqrt{n}$ for a sufficiently large constant c_4 and, since $K \geq c\sqrt{n}\log n$, we obtain a tail probability of $e^{-\Omega(c_4 c \log n)}$ with the implicit constant in the last Ω being independent of c_4 and c. Hence, by choosing the constant c from the theorem statement, we have a high-probability result for the hitting time T when the potential Φ_t becomes at most 9216.

When $\Phi_t \leq 9216$, the all-ones string is sampled with at least constant probability, denoted by $c_5 > 0$, assuming the worst case of as many frequencies as possible at the extremal values $\{1-1/n, 1/4\}$ and using well-known arguments on majorization and Schur-convexity. Next we consider a phase of length $c_6 \log n$ after time T, where $c_6 > 0$ is a constant chosen later, and bound the increase of potential in the phase using tailored arguments since the estimate from Corollary 1 would be too weak. We assume that $\Phi_t \leq 2 \cdot 9216$ throughout the phase and will justify later that the bound holds with high probability. Since each frequency p_i changes by $\pm 1/K$ per iteration with probability at most $2p_{t,i}(1-p_{t,i}) \leq 2(1-p_{t,i})$ independently of the other frequencies, the expected number of frequencies decreased is bounded from above by $\sum_{i=1}^{n} 2(1-p_{t,i}) = 2\Phi_t \leq 4 \cdot 9216$. Hence, the expected increase of potential is at most $4 \cdot 9216/K$. By Chernoff bounds, the increase is at most $(\log^2 n)/K$ with probability $1-2^{-\Omega(\log^2 n)}$, and by a union bound over the phase, the total increase is at most $(c_6 \log n)(\log^2 n)/K = (c_6 \log^3 n)/K \leq 9216$ with probability $1-2^{-\Omega(\log^2 n)}$, where we used our assumption on K. Hence, it holds that $\Phi_t \leq 9216 + 9216 = 2 \cdot 9216$ throughout the phase with probability $1-2^{-\Omega(\log^2 n)}$. Assuming this to happen, the probability of sampling the optimum is at least c_7 in every step of the phase for another constant c_7 depending only on c_5. Hence, for any constant c_6 controlling the phase length, the total success probability in the phase is at least $1-(1-c_7)^{c_6 \log n} \geq 1-e^{-c_8 c_6 \log n}$, where c_8 is a constant that depends on neither the constant c from the theorem statement nor the constant c_6. This proves that the global optimum is found in $O(K\sqrt{n})$ steps with high probability.

Analysis of genetic drift The previous considerations assumed that no significant genetic drift occurs, i.e., that all frequencies are bounded from below by $1/4$. For the case of OneMax, Lemma 9 bounds the probability that at least one frequency drops below $1/3$ (*not* $1/4$) in $O(K\sqrt{n})$ steps by n^{-c_9} for any constant $c_9 > 0$, choosing the constant $c > 0$ from the theorem statement large enough. Since sampling from the gap is superpolynomially unlikely if P_t is outside the critical region, we only need to study the impact of the time in the critical region on genetic drift, which we bounded by $O(K \log^4 n)$ with high probability further above. Pessimistically assuming that all steps where an individual is sampled in the gap decrease the frequency by $1/K$ and recalling that such steps happen with probability $O(1/\log^5 n)$, we have an accumulated decrease of $O((K \log^4 n)/(K \log^5 n)) \leq 1/12$ with high probability (assuming n large enough). Hence, by adjusting the starting frequency by $-1/12$, we obtain from the second statement of Lemma 9 for $a = 5/12$ that no frequency ever drops below $1/4$ with high probability. □

We have seen that the sampling variance of $\Theta(n)$ used in the early phases of optimization allows the cGA to overcome fitness gaps of asymptotic size almost $\sqrt{n}$. By contrast, if the gap is adjacent to the all-ones string like in the classical Jump function, then only logarithmically large gaps can be overcome in polynomial time, which follows from the lower bound on the runtime of the cGA given in Doerr [9].

We believe that we can improve the bound $O(\sqrt{n}/\log^5 n)$ on the gap size from Theorem 5 by a polylogarithmic factor by narrowing down the critical region and using drift analysis with tail bounds to derive a better bound on the time until the critical region from the proof is left. However, we leave a rigorous proof as subject for future work.

The analysis leading to the proof of Theorem 5 will also be crucial when studying the cGA on the JumpOffsetSpike function in Section 5.

4. The cGA samples fairly while following a fitness gradient

After having treated the JumpOffset function where the fitness gap is shifted compared to the original Jump, but the all-ones string is still the global optimum, we now prepare for an analysis of Jump functions with different location of the global optima, i.e., the JumpOffsetSpike function introduced in Section 2. In Section 5, we will obtain the surprising result that the cGA can find the optimal "spike" in the gap on such functions efficiently whereas the majority-vote crossover-based algorithms are inefficient. This result is based on the fact that the cGA has a high probability of sampling all middle fitness levels of the OneMax function, and this property will also turn out useful on JumpOffsetSpike.

We recall that the expected runtime of the cGA on OneMax is known to be $O(K\sqrt{n})$ for $K \geq c\sqrt{n}\log n$ with high probability, and this bound holds in expectation as well [43]. For an optimal parameter setting, this is $O(n \log n)$ and competitive with simple EAs. Clearly, on average over all n values, every OneMax-value is sampled an expected number of $\Theta(\log n)$ times during the optimization process. We now show the much stronger result that every number of ones in the interval $[n/2 + \epsilon n, n - \epsilon n]$ for an arbitrary constant $\epsilon > 0$ is sampled with high probability. This result is interesting in its own right, but, as mentioned above, also crucial for the analysis in the following section.

Theorem 6. *Let $K \geq c\sqrt{n}\log n$ for a sufficiently large constant $c > 0$ and $K = \text{poly}(n)$. Let $M \in [n/2 + \epsilon n, n - \epsilon n]$ for an arbitrary constant $\epsilon > 0$. Then the cGA, under the well-behaved frequencies assumption, with parameter K, running on the function* OneMax, *will sample a search point with M one-bits in $O(K\sqrt{n})$ steps with high probability.*

Proof. The main idea is to carefully analyze the potential $P_t = \sum_{i=1}^{n} p_{t,i}$ introduced in (4), which equals the expected number of one-bits sampled in the two offspring of the cGA. We will show that P_t stays in the vicinity of the value M for a sufficiently long time, allowing a high probability of sampling a search point with M one-bits thanks to Lemmas 3 and 4.

We recall that $P_t = n - \Phi_t$ for the alternative potential function Φ_t introduced in (3), and will use both P_t and Φ_t in our analysis, depending on which one is more convenient to use.

Throughout this proof, we consider an epoch of length $c_3 K\sqrt{n}$ for a sufficiently large constant $c_3 > 0$, which is chosen later (note that the constants c_1 and c_2 are the same as in Lemma 3, while c_3 and other constants in this proof may differ from homonymous variables appearing in earlier proofs). We assume that within the epoch, all frequencies are at least 1/3 (using Lemma 9) and that a potential $P_t \geq n - 9216$ is generated, according to Theorem 5, which applies to OneMax as well (if $m = 0$). The probability of the opposite happening is $e^{-\Omega(c \log n)}$ if c is large enough. This gives a term of $e^{-\Omega(c \log n)}$ for the total failure probability; further terms will follow.

Let the stopping time T be the first time t where $P_t \geq M$. We note that $P_0 = n/2$. Moreover, for fixed $t \geq 0$, we have $P_{t+1} - P_t \geq (\sqrt{n}\log^2 n)/K$ with probability $e^{-\Omega(\log^2 n)}$, which follows from Corollary 1 for $k = 1$ and $\lambda = \sqrt{n}\log^2 n$ (giving a crude estimate sufficient for our purposes). Therefore, we have $P_{s+1} - P_s \leq (\sqrt{n}\log^2 n)/K = O(\log n)$ for all $s \in [0, U]$ where $U = \text{poly}(n)$, with probability $1 - Ue^{-\Omega(\log^2(n))} = 1 - e^{-\Omega(\log^2(n))}$. Hence, $P_T \leq M + O(\log n)$ with probability $1 - e^{-\Omega(\log^2 n)}$.

We now consider a phase of $c_4\sqrt{n}\log n$ steps starting at time T, where $c_4 > 0$ is a constant chosen later, and denote by $T' := T + c_4\sqrt{n}\log n$ the end of the phase. Along with Corollary 1 for $k = c_4\sqrt{n}\log n$ and $\lambda = c_4 n \log n$, with probability $1 - e^{-\Omega(\sqrt{n}/\log n)} - e^{-\Omega(\log^2 n)}$, $P_{T'} \leq M + O(\log n) + (2c_4 n \log n)/K \leq M + 3(c_4/c)\sqrt{n}$ holds for n sufficiently large. This bounded potential has implications on the sampling variance, more precisely we have for $\sigma_t^2 = \sum_{i=1}^n p_{t,i}(1 - p_{t,i})$ that $\sigma_t^2 \geq \Phi_t/3 = (n - P_t)/3$ since all $p_{t,i} \geq 1/3$. Hence, since $M \leq (1 - \epsilon)n$, we have $\sigma_t \geq \sqrt{n - M - 3(c_4/c)\sqrt{n}} \geq \sqrt{\epsilon n/2}$ for all $t \in [T, T']$ and n large enough. With c_1 being the constant from Lemma 3, we choose c large enough such that

$$c_1\sqrt{\epsilon n/2} > \frac{3c_4}{c}\sqrt{\epsilon n/2} \tag{9}$$

holds. Hence, for all $t \in [T, T']$ there is a probability of at least c_2 (where c_2 stems from Lemma 3) that the cGA samples a search point with less than M one-bits and, analogously, also a probability of at least c_2 of sampling more than $P_t + 3(c_4/c)\sqrt{\epsilon n/2}$ one-bits. Note that we can assume $c_2 < 1/2$; if the actual tail probability for the considered deviation from the expected value is larger, we can simply consider a larger deviation since the tail probabilities decrease only by $O(1/\sigma_t) = O(1/\sqrt{n})$ per level of one-bits according to Lemma 2.

We now apply Lemma 4 with $\ell = u = c_2$ and obtain $\ell + u < 1$, $k_\ell < M$, and $k_u > M$. Hence, the probability of sampling M one-bits at any time $t \in [T, T']$ is at least $c_5/\sigma_t \geq c_5/\sqrt{\epsilon n/2}$ for the constant $c_5 > 0$ implicit in the expression $\Omega(1/\sigma)$ in Lemma 4. The probability of never sampling M ones in the phase is therefore bounded from above by

$$\left(1 - \frac{c_5}{\sqrt{\epsilon n/2}}\right)^{c_4\sqrt{n}\log n} \leq e^{-\Omega(c_4 \log n)},$$

where the implicit constant in the $\Omega()$ only depends on c_5 and ϵ. Now, by choosing c_4 large enough and, in turn also c large enough to satisfy (9), we have a total failure probability of $e^{-\Omega(c \log n)} + e^{-\Omega(c_4 \log n)} + e^{-\Omega(\log^2 n)}$ according to the definition of high probability. □

We think that it is possible to generalize Theorem 6 to all levels $M \in [n/2 + \epsilon n, n]$; however, this requires a more careful analysis of the typical sampling variance when P_t approaches M. In particular, if $M = n - o(n)$, then this variance will typically be $o(n)$ as well.

5. The cGA samples fairly within fitness gaps

We now consider the final of the example functions central for this paper, more precisely the function JumpOffsetSpike_m having the fitness gap shifted to $[3n/4, 3n/4 + m]$ ones-bits and the global optimum in the middle of this gap at $3n/4 + m/2$ one-bits. First, we show negative results for the algorithms using majority-vote crossover, confirming our claim that these are too much tailored towards creating the all-ones string. Afterwards, we show that the cGA is very efficient on this function.

Theorem 7. *Consider the* JumpOffsetSpike *function for $m < (3/32 - \epsilon)n$, where $\epsilon > 0$ is an arbitrarily small constant. With probability at least $1 - e^{-\Omega(n)}$, the hybrid GA (Algorithm 1) either stops prematurely or finds the all-ones string before the global optimum of this function and cannot create the global optimum afterwards.*

Similarly, consider the voting algorithm (Algorithm 2) with parameter $\mu \geq 32(c+1)n \log n$ as proposed in Rowe and Aishwaryaprajna [41] and assume furthermore that $\mu = \text{poly}(n)$. With probability at least $1 - O(n^{-c})$, it finds the all-ones string before the global optimum of the above-mentioned function and cannot create the global optimum afterwards.

Proof. We start with the proof for the voting algorithm. As analyzed in Rowe and Aishwaryaprajna [41, Theorem 3.2], the algorithm will with probability at least $1 - n^{-c}$ output the all-ones string as proposed solution if run on OneMax. We observe that the μ uniform samples made by the algorithm with overwhelming probability all fall into the case corresponding

to a value $m+|x|_1$ of the fitness function since sampling at least $3n/4$ one-bits has probability $2^{-\Omega(n)}$ according to Chernoff bounds along with a union bound over the $\mu = poly(n)$ samples. Since only the ranking of the search points matters, the behavior of the algorithm is equivalent to OneMax in this case. Hence, the proof of Theorem 3.2 still applies with an overall probability at least $1-O(n^{-c})-2^{-\Omega(n)} = 1-O(n^{-c})$ of outputting the all-ones string as proposed solution. By definition, the algorithm stops at this point.

We now turn to the hybrid GA. With probability $1-2^{-\Omega(n)}$ according to Chernoff bounds, its initial population consists of individuals with less than $3n/4$ one-bits, only. Then the subsequent next-ascent hillclimbing will create a population uniformly distributed on the level of $3n/4$ one-bits. Since $m \geq 4$ is assumed in the definition of JumpOffsetSpike, improving from the level of $3n/4$ one-bits requires at least two bits to be flipped simultaneously. Therefore, random ascent hillclimbing stops at this point and a majority-vote crossover of the three individuals is performed.

We proceed by building upon the simplified analysis of the hybrid GA presented in Rowe and Aishwaryaprajna [41]. The three individuals all have $n/4$ zero-bits and their location is independent of each other among these individuals. Hence, for each position the probability that majority-vote crossover creates a one-entry is, using $k = 3n/4$,

$$3\frac{n-k}{n}\left(\frac{k}{n}\right)^2 + \left(\frac{k}{n}\right)^3 = \frac{27}{32}.$$

Denoting by $Z_i \in \{0,1\}$ the indicator random variable for the event that majority-vote crossover creates a one-entry at bit i, where $1 \leq i \leq n$, we obtain from Lemma 6 that the random variables Z_i are negatively correlated. Using the Chernoff bound from Doerr [11, Th. 1.10.23], we have that $\sum_{i=1}^{n} Z_i > (27/32-\epsilon)n \geq 3n/4+m$ with probability at least $1-2^{-\Omega(n)}$. If this happens, the subsequent hillclimbing (if allowed) will work on the part $m+|x|_1$ of the function only, and reach the all-ones string. From there no other search points can be accepted. □

We now turn to another major result of this work, combining the ability of the cGA to cross large fitness gaps (Theorem 5) up to size $\tilde{O}(\sqrt{n})$ and the fair sampling property demonstrated in Theorem 6. In the proof of the following theorem, some arguments are only sketched when they appear in detail in the proof of Theorem 6.

Theorem 8. *Let $K \geq \lceil c\sqrt{n}\log n\rceil$ for a sufficiently large constant $c > 0$ and $K = \text{poly}(n)$. Let $m = O(\sqrt{n}/\log^5 n)$. Then the cGA, under the well-behaved frequencies assumption, with parameter K will find the global optimum of JumpOffsetSpike_m in $O(K\sqrt{n})$ steps with high probability. The last bound is $O(n\log n)$ for $K = \lceil c\sqrt{n}\log n\rceil$.*

Proof. Since the cGA will behave identically on JumpOffset and JumpOffsetSpike before the optimum at $3n/4+m/2$ one-bits is sampled, we pretend that the algorithm runs on JumpOffset_m and assume that all frequencies are bounded from below by $1/4$ before the optimum is found. Again we inspect the dynamics of the potential $P_t = \sum_{i=1}^{n} p_{t,i}$ and its inverse $\Phi_t = n - P_t$ closely. Following the proof of Theorem 5, we note that P_t has a drift of $\sqrt{\Phi_t}/(24K)$ for $\Phi_t \geq 9216$. Hence, we can use additive drift (Th. 2.3.1 in Lengler [31]; alternatively Th. 2 in the special case that $h(x)$ is uniformly bounded from below by some δ for all $x \geq x_{\min}$) to analyze the first point of time T where $P_t \geq 3n/4+m/2$. Arguing like in the proof of Theorem 6, we have $P_T \leq 3n/4+m/2+O(\log n)$ with probability $1-e^{-\Omega(\log^2 n)}$, which we assume to happen. So, we can handle the possible overshooting of the target $3n/4+m/2$ in the same way as in the additive drift analysis in the proof of Theorem 5 on page 30 and obtain $\mathrm{E}(T) = O(K\sqrt{n})$.

Note that our bound on P_T implies $\sigma_T = \Omega(\sqrt{P_T}) = \Omega(\sqrt{n})$. Considering a phase of $c_4\sqrt{n}\log n$ steps starting at time T for some appropriate constant $c_4 > 0$ and denoting by T' the end of the phase, we observe that the potential changes by no more than $O((\sqrt{n}\log n)\sqrt{n}/K) = O(\sqrt{n})$ in the phase with superpolynomially large probability, thanks to Corollary 1. Note that this lemma does not make any assumption on the underlying fitness function and hence holds for periods where the cGA samples in the gap as well. Therefore, there is a probability of $\Omega(1/\sqrt{n})$ of sampling a search point with $3n/4+m/2$ ones-bits, i.e., the global optimum, for each step of the phase. Choosing the constants, in particular c, appropriately, the probability of not sampling the global optimum in the phase is bounded from above by $n^{-\Omega(c)}$ and the total failure probability by $n^{-c'}$ for a constant c' we can choose ourselves. Note that the choice of c' affects the minimum size of the constant c. □

To summarize, the cGA displays an interesting search trajectory on the JumpOffsetSpike function that "filters out" deceptive information and thus is able to approach the all-ones string despite the reversed fitness gradient in the gap region. Such effects are also known for classical genetic algorithms and termed as implicit "low-pass filter" in [37]. However, it is worth pointing out that the results of Theorem 6 and Theorem 8 only show that the cGA is likely to sample from each *Hamming level* $m \in [n/2+\epsilon n, n-\epsilon n]$ at least once. If the set of global optima was only a small subset of a particular Hamming level, then the cGA could be likely to miss these optima as well as discussed in the following.

Consider, e.g., a modified JumpOffset function where search points having exactly $3n/4$ one-bits are not treated symmetrically. The point $1^{3n/4}0^{n/4}$ is globally optimal and the fitness among the strings with $3n/4$ one-bits is monotone decreasing in the Hamming distance to the global optimum; formally, we suggest

$$f_m(x_1,\dots,x_n) = \begin{cases} m + |x|_1 & \text{if } |x|_1 < 3n/4 \text{ or } |x|_1 > 3n/4 + m, \\ 2n + m + 1 - H(x, 1^{3n/4}0^{n/4}) & \text{if } |x|_1 = 3n/4, \\ 3n/4 + m - |x|_1 & \text{otherwise,} \end{cases}$$

where $H(\cdot,\cdot)$ denotes the Hamming distance.

Then a hillclimber like the (1+1) EA could efficiently reach the global optimum using two-bit flips within the Hamming level of $3n/4$ one-bits; however, we think that the cGA would reach the all-ones string first then need superpolynomial time to sample the global optimum.

6. Gap sizes that the cGA cannot cross efficiently

We have seen in Theorem 5 that the cGA is able to overcome gap sizes $m = O(\sqrt{n}/\log^5 n)$. This result is essentially tight, up to polylogarithmic factors. In this section, for proof reasons, we switch to the cGA without borders, i.e., the variant where the minimization of frequencies with $1 - 1/n$ and maximization with $1/n$ is omitted. We will consider gap sizes $m = \Omega(\sqrt{n}\log^2 n)$ and show that the cGA without borders cannot overcome efficiently.

Theorem 9. *The cGA without borders and arbitrary $K = n^{O(1)}$ on the fitness function* JumpOffset *with $m = \Omega(\sqrt{n}\log^2 n)$ does not sample the all-ones string in time $n^{c\log n}$, where $c > 0$ is a constant, with probability $1 - n^{-\Omega(\log n)}$.*

Note that the theorem does not demand well-behaved frequencies in the sense of Section 2.4.

The idea of the proof is that the large gap size makes it likely that the cGA samples from the gap and experiences higher fitness from individuals with a lower number of one-bits. This implies a decrease of the potential P_t, which makes it unlikely to sample search points to the right of the gap, i.e., with more than $3n/4 + m$ one-bits.

Proof of Theorem 9. We once again use the potential $P_t = \sum_{i=1}^{n} p_{t,i}$ introduced in Section 2.4. Again, the gap region is denoted by $G = [3n/4, 3n/4 + m]$, and we will distinguish between different cases with respect to the middle third of the gap region $M = [a, b]$, where $a = 3n/4 + m/3$ and $b = 3n/4 + 2m/3$. If $P_t < a$, then the probability of sampling an individual with more than b one-bits is $n^{-\Omega(\log n)}$ by Lemma 7 since $m = \Omega(\sqrt{n}\log^2 n)$. Moreover, applying Corollary 1 with $k = 1$, $\lambda = m/3$, and using $K \geq 2$ from the definition of the cGA, we have that $P_{t+1} \leq b$ with probability $1 - n^{-\Omega(\log n)}$.

We now consider the case that $P_t \in M$, i.e., $a \leq P_t \leq b$, which means that the expected number of one-bits of an offspring of the cGA is within M. Moreover, the probability that the number is in $[0, n] \setminus G$, i.e., outside the gap, is $n^{-\Omega(\log n)}$ again by Lemma 7. Let us study the two offspring $x \in \{0,1\}^n$ and $y \in \{0,1\}^n$ sampled in a generation and assume that both $|x|_1 \in G$ and $|y|_1 \in G$. The cGA arranges the two offspring according to fitness so that JumpOffset$(x) \geq$ JumpOffset(y). Since the offspring are in the gap, we have $|x|_1 \leq |y|_1$. Hence, the number of frequencies decreased by $1/K$ in the following update of frequencies must be at least as big as the number of frequencies increased. Since the algorithm does not have borders on frequencies, this implies $P_{t+1} \leq P_t$.

Altogether, up to events of probability $n^{-\Omega(\log n)}$, the algorithm displays the following behavior: if $P_t < a$, then the offspring of the cGA have at most b one-bits, and $P_{t+1} \leq b$ holds. If $P_t \in M$, then $P_{t+1} \leq P_t \leq b$. When the potential is at most b, the offspring have at most $3n/4 + m$ one-bits, i.e., are not optimal, with probability at least $1 - n^{-\Omega(\log n)}$. Since an event of probability $n^{-\Omega(\log n)}$ does not happen in a phase of $n^{c\log n}$ steps, where c is a sufficiently small constant, with probability $1 - n^{-\Omega(\log n)}$ thanks to a union bound, this proves the theorem. □

It is interesting that both the result of Theorem 9, and the lower bound for the classical Jump function in Doerr [9], which was proved using an alternative drift theorem, do not assume a minimum value for K. Hence, the analysis also holds in the regime with strong genetic drift corresponding to $K = o(\sqrt{n}\log n)$ that leads to a different internal behavior than the case $K = \Omega(\sqrt{n}\log n)$ assumed in, e.g., Theorem 5. However, we emphasize again that Theorem 9 assumes that the cGA does not use frequency borders. This assumption is caused by the fact that we cannot really control genetic drift in either direction. If search points are sampled outside the gap, frequencies tend to increase and reaching the lower border is unlikely enough for large enough K; however the situation is reversed when both offspring are in the gap, which is exactly the situation considered in proof of the theorem. Altogether, without a more detailed understanding of the number of frequencies at the borders, a negative drift within the gap cannot be established easily. Interestingly, the lower bound for the cGA on the classical Jump function in Doerr [9] does not suffer from these issues since the gap is adjacent to the all-ones string there and the drift region is set so close to the all-ones string that the effect of frequencies at the lower border is negligible. Hence, that result applies also to the cGA with borders; however, it cannot be transferred to JumpOffset easily due to the shift of the gap.

It is a subject for future work to generalize Theorem 9 to the cGA with borders. Nevertheless, all positive results proved for the cGA with borders in this paper (i.e., Theorem 5, Theorem 6, and Theorem 8) would also hold for the cGA without borders.

7. Experiments

We implemented the cGA in the C programming language using the WELL512a random number generator and determined the average number of iterations until sampling the optimum of JumpOffset for different values of n and K, averaged over 1000 runs. We used $n = (10i)^2$ for $i = \{1, \dots, 5\}$ and also for $n = 200$ since it is cheap to do experiments on such small n, i.e., we used $n \in \{100, 200, 400, 900, 1600, 2500\}$. For each such n, we considered three different values of K, more precisely $K = \lfloor c\sqrt{n}\ln n \rfloor$ for $c \in \{3, 9, 27\}$ inspired by the requirement $K \geq c\sqrt{n}\ln n$ from Theorem 5. (Note that our implementation rounded down K to the nearest integer, whereas most theorems in this paper rounded K up; we ignore these minimal discrepancies here.) Based on preliminary experimental trials, we chose a timeout of $20K\sqrt{n}\ln n$ iterations for the cGA and considered runs exceeding this threshold as unsuccessful.

We then recorded the average runtimes with the different parameter settings for n and K and let the jump size m increase from 2 in steps of size 1 until all three settings of K lead to unsuccessful runs in at least 50% of the repetitions. Figure 5 displays the experimental data collected. For each value of n there is one diagram displaying three curves corresponding to $c = 3$ (blue), $c = 9$ (red) and $c = 27$ (brown). Missing data points indicate that there were at least 50% unsuccessful runs for the corresponding value of K.

Looking for each n into the largest m that allows successful runs for at least one value of K and the majority of the 1000 runs, we see in the following table that a quadratic increase of n leads to a linear increase of the largest successful m (in fact, for this particular set of experiments we even have that the largest successful m equals $\sqrt{n}/2 - 1$). This is in line with our theoretical statements, which show that $m = \tilde{\Theta}(\sqrt{n})$ is the largest order of magnitude where efficient runtimes are possible.

n	largest m
100	4
400	9
900	14
1600	19
2500	24

Finally, we also investigated the "fair sampling" property from Section 4 experimentally. We ran the cGA with $n = 2500$, $K = \lfloor 27\sqrt{n}\ln n \rfloor = 10562$ on JumpOffset first with $m = 3$ and then with $m = 15$ and recorded the average number of times that a search point with i one-bits, where $i = 0, \dots, n$, was sampled before the optimum was created for the first time. The average was taken over 1000 runs. Figure 6 shows that all Hamming levels from level roughly $n/2$ are sampled frequently, with an average count of about 1000 for the majority of the levels, followed by a steep increase to 4000 around n. The final drop visible in the figure is due to the fact that the algorithm is stopped with the first sample of the all-ones string. We also notice the small "bump" around $3n/4$, indicating that samples slightly before and around the gap level occur a bit more frequently than for the surrounding levels. This bump is more pronounced for $m = 15$ than for $m = 3$, indicating the increased time the potential spends around the gap; otherwise the curves for the two different jump sizes are basically identical.

8. Conclusions

We have considered two algorithms using majority-vote crossover and the simple estimation-of-distribution algorithm cGA on modified jump functions. If the fitness valley of Jump is shifted to roughly the middle of the typical search trajectory, the crossover-based algorithm can still efficiently overcome such valleys, which is a consequence of the majority vote leading to the all-ones string. The cGA can overcome gaps of size up to almost $\sqrt{n}$ in this case, which is much larger than the gap size $O(\log n)$ required for efficient optimization of the classic jump function. If the global optimum is within the shifted fitness gaps instead of the all-ones string, the crossover-based algorithms fail with high probability to find the optimum efficiently. This stands in sharp contrast to the cGA, which still can sample such a global optimum efficiently for gap sizes almost $\sqrt{n}$. To prove this result, we have established a property called *fair sampling* of fitness levels, using a careful mathematical analysis of the underlying sampling process.

This work raises a number of open questions. For example, one could try to determine more precisely the gap size where the optimization time of the cGA turns from polynomial (with high probability) to exponential with high probability. So far these two regimes are only separated by polylogarithmic factors, and the proof of the lower bound only applies to the cGA without borders. Moreover, we have only analyzed the case that the fitness gap starts at $3n/4$ one-bits. A generalization of the cGA analysis to an arbitrary starting point of the gap, as partially considered for mutation-based algorithms in Bambury et al. [4], would be interesting. At the same time, extensions of the crossover-based algorithms, e.g., a voting algorithm that repeats sampling search points from majority vote until a stopping criterion is fulfilled, could be considered.

Another aspect to study is the convergence of the sampling distribution to the set of optimal solutions. Our results for JumpOffsetSpike show that the cGA is able to sample an optimum located in a set $S := \{x \mid |x|_1 = m\}$ being far from the

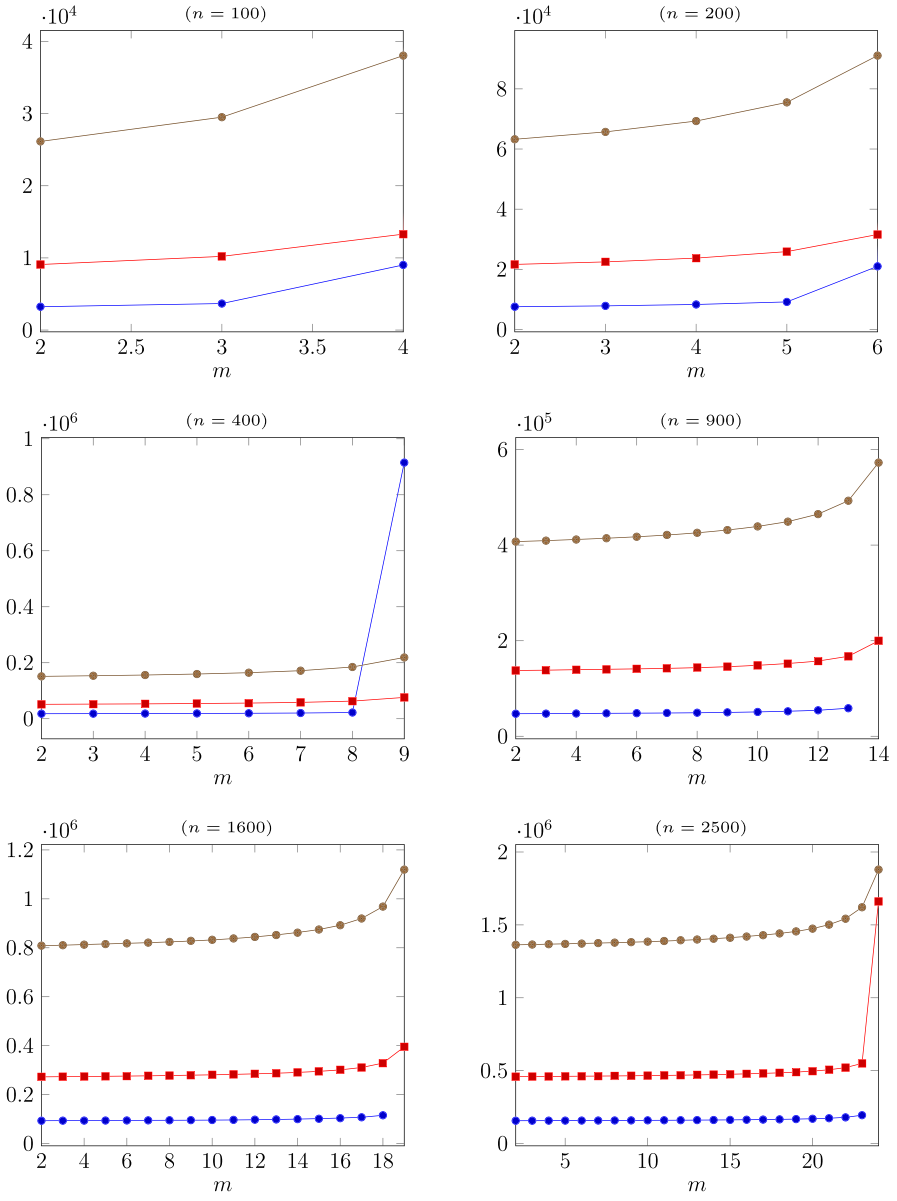

Fig. 5. Empirical runtime of the cGA on JumpOffset for growing m, $n = 100, 200, 400, 900, 1600, 2500$ in left-to-right then top-to-bottom order; each value of n tried with $K = 3\sqrt{n}\ln n$ (blue), $K = 9\sqrt{n}\ln n$ (red) and $K = 27\sqrt{n}\ln n$ (brown). (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

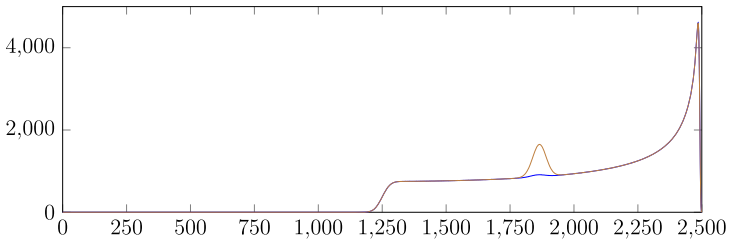

Fig. 6. No. of samples falling into the Hamming levels $i = 0, \ldots, 2500$ when running the cGA $n = 2500$, $K = 10562$ on JumpOffset with $m = 3$ (blue) and $m = 15$ (brown).

all-ones string in polynomial time with high probability; however, it does not show that in the long run the distribution is concentrated on the points in S. In fact, we do not think this is true in general. Similarly, the analyses of the cGA on the original Jump function [10] do not show convergence of the distribution to the all-ones string but exploit that the algorithm is able to sample the optimum when the distribution is sufficiently close to the one setting all frequencies to their maximum. It is a subject for further research to include convergence aspects of the sampling distribution in the runtime analysis of EDAs.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work was supported by a grant from the Danish Council for Independent Research (DFF-FNU 8021-00260B). The author thanks the anonymous reviewers for their constructive comments and suggestions which helped to improve the manuscript.

References

[1] D. Ackley, A Connectionist Machine for Genetic Hillclimbing, Springer, 1987.
[2] D. Antipov, B. Doerr, V. Karavaev, A rigorous runtime analysis of the (1 + (λ, λ)) GA on jump functions, Algorithmica 84 (2022) 1573–1602.
[3] J.B. Baillon, R. Cominetti, J. Vaisman, A sharp uniform bound for the distribution of sums of Bernoulli trials, Comb. Probab. Comput. 25 (2016) 352–361.
[4] H. Bambury, A. Bultel, B. Doerr, An extended jump function benchmark for the analysis of randomized search heuristics, in: Proc. of GECCO '21, ACM Press, 2021, pp. 1124–1132.
[5] R. Benbaki, Z. Benmar, B. Doerr, A rigorous runtime analysis of the 2-MMAS$_{\text{ib}}$ on jump functions: ant colony optimizers can cope well with local optima, in: Proc. of GECCO '21, ACM Press, 2021, pp. 4–13.
[6] D. Corus, P.S. Oliveto, D. Yazdani, On the runtime analysis of the Opt-IA artificial immune system, in: Proc. of GECCO '17, ACM Press, 2017, pp. 83–90.
[7] D. Corus, P.S. Oliveto, D. Yazdani, Fast artificial immune systems, in: Proc. of PPSN '18, Springer, 2018, pp. 67–78.
[8] D. Dang, T. Friedrich, T. Kötzing, M.S. Krejca, P.K. Lehre, P.S. Oliveto, D. Sudholt, A.M. Sutton, Escaping local optima using crossover with emergent diversity, IEEE Trans. Evol. Comput. 22 (2018) 484–497.
[9] B. Doerr, An exponential lower bound for the runtime of the compact genetic algorithm on jump functions, in: Proc. of FOGA '19, ACM Press, 2019, pp. 25–33.
[10] B. Doerr, A tight runtime analysis for the cGA on jump functions: EDAs can cross fitness valleys at no extra cost, in: Proc. of GECCO '19, ACM Press, 2019, pp. 1488–1496.
[11] B. Doerr, Probabilistic tools for the analysis of randomized optimization heuristics, in: B. Doerr, F. Neumann (Eds.), Theory of Evolutionary Computation – Recent Developments in Discrete Optimization, Springer, 2020, pp. 1–87.
[12] B. Doerr, The runtime of the compact genetic algorithm on jump functions, Algorithmica 83 (2021) 3059–3107, Journal paper combining the above GECCO 19 and FOGA 19 papers by the same author.
[13] B. Doerr, H.P. Le, R. Makhmara, T.D. Nguyen, Fast genetic algorithms, in: Proc. of GECCO '17, ACM Press, 2017, pp. 777–784.
[14] B. Doerr, W. Zheng, Sharp bounds for genetic drift in estimation of distribution algorithms, IEEE Trans. Evol. Comput. 24 (2020) 1140–1149.
[15] S. Droste, A rigorous analysis of the compact genetic algorithm for linear functions, Nat. Comput. 5 (2006) 257–283.
[16] S. Droste, T. Jansen, I. Wegener, On the analysis of the (1+1) evolutionary algorithm, Theor. Comput. Sci. 276 (2002) 51–81.
[17] S. Droste, T. Jansen, I. Wegener, Upper and lower bounds for randomized search heuristics in black-box optimization, Theory Comput. Syst. 39 (2006) 525–544.
[18] X. Fan, I. Grama, Q. Liu, Hoeffding's inequality for supermartingales, Stoch. Process. Appl. 122 (2012) 3545–3559.
[19] T. Friedrich, T. Kötzing, M.S. Krejca, S. Nallaperuma, F. Neumann, M. Schirneck, Fast building block assembly by majority vote crossover, in: Proc. of GECCO '16, ACM Press, 2016, pp. 661–668.
[20] T. Friedrich, T. Kötzing, M.S. Krejca, A.M. Sutton, The compact genetic algorithm is efficient under extreme Gaussian noise, IEEE Trans. Evol. Comput. 21 (2017) 477–490.
[21] T. Friedrich, F. Quinzan, M. Wagner, Escaping large deceptive basins of attraction with heavy-tailed mutation operators, in: Proc. of GECCO '18, ACM Press, 2018, pp. 293–300.
[22] L.J. Gleser, On the distribution of the number of successes in independent trials, Ann. Probab. 3 (1975) 182–188.
[23] G.R. Harik, F.G. Lobo, D.E. Goldberg, The compact genetic algorithm, IEEE Trans. Evol. Comput. 3 (1999) 287–297.
[24] W. Hoeffding, On the distribution of the number of successes in independent trials, Ann. Math. Stat. 27 (1956) 713–721.
[25] H. Hwang, C. Witt, Sharp bounds on the runtime of the (1+1) EA via drift analysis and analytic combinatorial tools, in: Proc. of FOGA '19, ACM Press, 2019, pp. 1–12.
[26] T. Jansen, On the black-box complexity of example functions: the real jump function, in: Proc. of FOGA '15, ACM Press, 2015, pp. 16–24.
[27] T. Jansen, I. Wegener, The analysis of evolutionary algorithms - a proof that crossover really can help, Algorithmica 34 (2002) 47–66.
[28] D. Johannsen, Random combinatorial structures and randomized search heuristics, Ph.D. thesis, Saarland University, 2010.
[29] T. Kötzing, D. Sudholt, M. Theile, How crossover helps in Pseudo-Boolean optimization, in: Proc. of GECCO 2011, ACM Press, 2011, pp. 989–996.
[30] P.K. Lehre, C. Witt, Tail bounds on hitting times of randomized search heuristics using variable drift analysis, Comb. Probab. Comput. 30 (2021) 550–569.
[31] J. Lengler, Drift analysis, in: B. Doerr, F. Neumann (Eds.), Theory of Evolutionary Computation – Recent Developments in Discrete Optimization, Springer, 2020, pp. 89–131.
[32] J. Lengler, D. Sudholt, C. Witt, The complex parameter landscape of the compact genetic algorithm, Algorithmica 83 (2021) 1096–1137.
[33] A. Lissovoi, P.S. Oliveto, J.A. Warwicker, On the time complexity of algorithm selection hyper-heuristics for multimodal optimisation, in: Proc. of AAAI '19, AAAI Press, 2019, pp. 2322–2329.
[34] A.W. Marshall, I. Olkin, B.C. Arnold, Inequalities: Theory of Majorization and Its Applications, 2nd ed., Springer, 2011.

[35] B. Mitavskiy, J.E. Rowe, C. Cannings, Theoretical analysis of local search strategies to optimize network communication subject to preserving the total number of links, Int. J. Intell. Comput. Cybern. 2 (2009) 243–284.
[36] P.S. Oliveto, C. Witt, Improved time complexity analysis of the simple genetic algorithm, Theor. Comput. Sci. 605 (2015) 21–41.
[37] A. Prügel-Bennett, Benefits of a population: five mechanisms that advantage population-based algorithms, IEEE Trans. Evol. Comput. 14 (2010) 500–517.
[38] A. Rajabi, C. Witt, Self-adjusting evolutionary algorithms for multimodal optimization, in: Proc. of GECCO '20, ACM Press, 2020, pp. 1314–1322.
[39] A. Rajabi, C. Witt, Stagnation detection in highly multimodal fitness landscapes, in: Proc. of GECCO '21, ACM Press, 2021, pp. 1178–1186.
[40] A. Rajabi, C. Witt, Stagnation detection with randomized local search, in: Proc. of EvoCOP '21, Springer, 2021, pp. 152–168.
[41] J.E. Rowe, Aishwaryaprajna, The benefits and limitations of voting mechanisms in evolutionary optimisation, in: Proc. of FOGA '19, ACM Press, 2019, pp. 34–42.
[42] D. Sudholt, How crossover speeds up building block assembly in genetic algorithms, Evol. Comput. 25 (2017) 237–274.
[43] D. Sudholt, C. Witt, On the choice of the update strength in estimation-of-distribution algorithms and ant colony optimization, Algorithmica 81 (2019) 1450–1489.
[44] D. Whitley, S. Varadarajan, R. Hirsch, A. Mukhopadhyay, Exploration and exploitation without mutation: solving the jump function in $\theta(n)$ time, in: Proc. of PPSN '18, Springer, 2018, pp. 55–66.
[45] C. Witt, Domino convergence: why one should hill-climb on linear functions, in: Proc. of GECCO '18, ACM Press, 2018, pp. 1539–1546.
[46] C. Witt, Upper bounds on the running time of the univariate marginal distribution algorithm on OneMax, Algorithmica 81 (2019) 632–667.
[47] C. Witt, On crossing fitness valleys with majority-vote crossover and estimation-of-distribution algorithms, in: Proc. of FOGA '21, ACM Press, 2021, pp. 1–15.