A Tight Runtime Analysis for the cGA on Jump Functions—EDAs Can Cross Fitness Valleys at No Extra Cost

Benjamin Doerr
École Polytechnique, CNRS, Laboratoire d'Informatique (LIX)
Palaiseau, France

ABSTRACT

We prove that the compact genetic algorithm (cGA) with hypothetical population size $\mu = \Omega(\sqrt{n} \log n) \cap \text{poly}(n)$ with high probability finds the optimum of any n-dimensional jump function with jump size $k < \frac{1}{20} \ln n$ in $O(\mu\sqrt{n})$ iterations. Since it is known that the cGA with high probability needs at least $\Omega(\mu\sqrt{n} + n \log n)$ iterations to optimize the unimodal OneMax function, our result shows that the cGA in contrast to most classic evolutionary algorithms here is able to cross moderate-sized valleys of low fitness at no extra cost.

Our runtime guarantee improves over the recent upper bound $O(\mu n^{1.5} \log n)$ valid for $\mu = \Omega(n^{3.5+\varepsilon})$ of Hasenöhrl and Sutton (GECCO 2018). For the best choice of the hypothetical population size, this result gives a runtime guarantee of $O(n^{5+\varepsilon})$, whereas ours gives $O(n \log n)$.

We also provide a simple general method based on parallel runs that, under mild conditions, (i) overcomes the need to specify a suitable population size, but gives a performance close to the one stemming from the best-possible population size, and (ii) transforms EDAs with high-probability performance guarantees into EDAs with similar bounds on the expected runtime.

CCS CONCEPTS

• **Theory of computation** → **Random search heuristics**;

KEYWORDS

Estimation-of-distribution algorithm; theory; runtime analysis

ACM Reference Format:
Benjamin Doerr. 2019. A Tight Runtime Analysis for the cGA on Jump Functions— EDAs Can Cross Fitness Valleys at No Extra Cost. In *Genetic and Evolutionary Computation Conference (GECCO '19), July 13–17, 2019, Prague, Czech Republic.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3321707.3321747

1 INTRODUCTION

While the mathematical analysis of evolutionary algorithms (EAs) has produced a plethora of insightful results in the last over 20 years, the rigorous understanding of estimation-of-distribution algorithms (EDAs) is much less developed [21]. Obviously, this is due to the highly complex stochastic processes that describe the runs of such algorithms. In consequence, despite significant efforts and deep results [14, 23, 24], not even the runtime of the compact genetic algorithm (cGA) on the OneMax benchmark function is fully understood (here we would argue that the cGA is the most simple EDA and that the unimodal OneMax function, counting the number of ones in a bit string, is the most simple optimization problem with unique global optimum). It is therefore not surprising that many questions which are well-understood for EAs are only started being understood for EDAs. One such question is how EDAs optimize objective functions that are not unimodal.

In the first and so far only runtime analysis of an EDA on a non-unimodal objective function, Hasenöhrl and Sutton [17] regard the optimization time of the cGA on the jump function class, which are unimodal apart from having a valley of low fitness of scalable size k around the global optimum. They show [17, Theorem 3.3] that, for a sufficiently large constant C and any constant $\varepsilon > 0$, the cGA with hypothetical population size at least $\mu \geq \max\{Cne^{4k}, n^{3.5+\varepsilon}\}$[1] with probability $1 - o(1)$ finds the optimum of any jump function with jump size at most $k = o(n)$ in $O(\mu n^{1.5} \log n + e^{4k})$ generations (which is also the number of fitness evaluations, since the cGA evaluates only two search points in each iteration).

This result is remarkable in that it shows that the cGA with the right choice of μ and for $k \geq 6$ is more efficient on jump functions than most evolutionary algorithms, who have a runtime of at least $\Omega(n^k)$, see Section 2.2.

There is one aspect in which the result of Hasenöhrl and Sutton is not yet perfect (and this is the motivation of this work). We note that even when choosing the smallest possible population size $\mu = n^{3.5+\varepsilon}$, the runtime guarantee becomes at least $\Omega(n^{5+\varepsilon})$. While clearly a polynomial runtime, and thus *efficient* in the classic complexity theory view, this is a runtime that is not practical in many applications (and we recall here that the target of the mathematical analysis of evolutionary algorithms is not to understand jump functions, but to derive from the analysis on simple test problems insight that extend to practically relevant problems). Also, this runtime guarantee is weaker than the $O(n^k)$ bound for simple mutation-based EAs such as the $(1 + 1)$ EA when $k \leq 5$. Hence one could feel that the result of Hasenöhrl and Sutton shows the superiority of EDAs rather for problem instances for which both the runtime of typical EAs and the performance guarantee for the cGA are prohibitively large. In a similar vein, one has to question if a practitioner would run the cGA with a hypothetical population size of more than $n^{3.5}$ when solving a problem defined over bit strings of length n.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
GECCO '19, July 13–17, 2019, Prague, Czech Republic
© 2019 Copyright held by the owner/author(s). Publication rights licensed to the Association for Computing Machinery.
ACM ISBN 978-1-4503-6111-8/19/07…$15.00
https://doi.org/10.1145/3321707.3321747

[1]In the paper, this is stated as minimum of the two terms, but from the proofs it is clear that it should be the maximum.

Our main result is that these potential weaknesses of the cGA are not real and that the cGA performs in fact much better than what the previous work shows. We prove rigorously that the cGA with hypothetical population size $\mu \geq K\sqrt{n}\log n$, K a sufficiently large constant, and μ polynomially bounded in n with high probability optimizes any n-dimensional jump function with jump size $k < \frac{1}{20}\ln n$ in only $O(\mu\sqrt{n})$ iterations. For the smallest admissible populations size $\mu = \Theta(\sqrt{n}\log n)$, this gives a runtime guarantee of $O(n\log n)$, a result that both overcomes the large runtime and the large required hypothetical population size of the previous result.

From a broader perspective our result shows that the cGA (and we expect similar result to hold for other EDAs) does not suffer from moderate-size valleys of low fitness. We recall that Sudholt and Witt [24] have shown that the cGA with any hypothetical population size (polynomial in n) with high probability needs $\Omega(\mu\sqrt{n}+n\log n)$ iterations to optimize the OneMax function. Hence our result shows that adding a valley of low fitness to the OneMax function does not worsen the asymptotic performance of the cGA as long as the fitness valley is smaller than $\frac{1}{20}\ln n$.

We may add that our work also makes some arguments of [17] more rigorous. In particular, we observe that the progress of the cGA cannot be estimated by taking the progress one would have when no fitness valley was present and correcting this estimate by inverting the progress with the probability that a search point is sampled in the fitness valley. This argument ignores the stochastic dependencies between the absolute value of the progress and the event that a solution in the gap is sampled. These dependencies are real and have a negative impact as discussed in more detail before Lemma 6.

We note that the approach of intentionally ignoring some dependencies to make a mathematical analysis tractable, often called mean-field analysis, is common in some scientific areas, most notably statistical physics, and has also been used in evolutionary computation, e.g., [26]. This approach, however, needs an additional justification, e.g., via specific experiments, why the omission of the dependencies should not change the matter substantially. In any case, such mean-field approaches do not lead to results fully proven in the mathematical sense. In this sense, we hope that our work also provides methods that help in future analyses of EDAs on non-unimodal optimization problems.

As a **side result**, triggered by the fact that we "only" show a bound that holds with high probability, but not a bound on the expected runtime, we provide a general approach to transform an EDA using a population size parameter μ into an algorithm that does not require the specification of such a parameter, but has a performance similar to the one of the EDA with optimally chosen parameter. This performance guarantee also holds for the expected runtime, even if for the EDA only a with-high-probability runtime guarantee is known.

2 PRELIMINARIES

2.1 The Compact Genetic Algorithm

The *compact genetic algorithm* (cGA) is an estimation-of-distribution algorithm (EDA) proposed by Harik, Lobo, and Goldberg [16] for the maximization of pseudo-Boolean functions $\mathcal{F} : \{0,1\}^n \to \mathbb{R}$. Being a univariate EDA, it develops a probabilistic model described by a frequency vector $f \in [0,1]^n$. This frequency vector describes a probability distribution on the search space $\{0,1\}^n$. If $X = (X_1,\dots,X_n) \in \{0,1\}^n$ is a search point sampled according to this distribution—we write

$$X \sim \text{Sample}(f)$$

to indicate this—then we have $\Pr[X_i = 1] = f_i$ independently for all $i \in [1..n] := \{1,\dots,n\}$. In other words, the probability that X equals some fixed search point y is

$$\Pr[X = y] = \prod_{i:y_i=1} f_i \prod_{i:y_i=0} (1 - f_i).$$

In each iteration, the cGA updates this probabilistic model as follows. It samples two search points $x^1, x^2 \sim \text{Sample}(f)$, computes the fitness of both, and defines $(y^1, y^2) = (x^1, x^2)$ when x^1 is at least as fit as x^2 and $(y^1, y^2) = (x^2, x^1)$ otherwise. Consequently, y^1 is the rather better search point of the two. We then define a preliminary model by $f' := f + \frac{1}{\mu}(y^1 - y^2)$. This definition ensures that, when y^1 and y^2 differ in some bit position i, the i-th preliminary frequency moves by a step of $\frac{1}{\mu}$ into the direction of y^1_i, which we hope to be the right direction since y^1 is the better of the two search points. The *hypothetical populations size* μ is used to control how strong this update is.

To avoid a premature convergence, we ensure that the new frequency vector is in $[\frac{1}{n}, 1 - \frac{1}{n}]^n$ by capping too small or too large values at the corresponding boundaries. More precisely, for all $\ell \leq u$ and all $r \in \mathbb{R}$ we define

$$\text{minmax}(\ell, r, u) := \max\{\ell, \min\{r, u\}\} = \begin{cases} \ell & \text{if } r < \ell \\ r & \text{if } r \in [\ell, u] \\ u & \text{if } r > u \end{cases}$$

and we lift this notation to vectors by reading it component-wise. Now the new frequency vector is $\text{minmax}(\frac{1}{n}\mathbf{1}_n, f', (1 - \frac{1}{n})\mathbf{1}_n)$.

This iterative frequency development is pursued until some termination criterion is met. Since we aim at analyzing the time (number of iterations) it takes to sample the optimal solution (this is what we call the *runtime* of the cGA), we do not specify a termination criterion and pretend that the algorithm runs forever.

The pseudo-code for the cGA is given in Algorithm 1. We shall use the notation given there frequently in our proofs. For the frequency vector f_t obtained at the end of iteration t, we denote its i-th component by $f_{i,t}$ or, when there is no risk of ambiguity, by f_{it}.

Well-behaved frequency assumption: For the hypothetical population size μ, we take the common assumption that any two frequencies that can occur in a run of the cGA differ by a multiple of $\frac{1}{\mu}$. We call this the *well-behaved frequency assumption*. This assumption was implicitly already made in [16] by using even μ in all experiments (note that the hypothetical population size is denoted by n in [16]). This assumption was made explicit in [14] by requiring μ to be even. Both works do not use the frequencies boundaries $\frac{1}{n}$ and $1-\frac{1}{n}$, so an even value for μ ensures well-behaved frequencies.

For the case with frequency boundaries, the well-behaved frequency assumption is equivalent to $(1 - \frac{2}{n})$ being an even multiple of the update step size $\frac{1}{\mu}$. In this case, $n_\mu = (1 - \frac{2}{n})\mu \in 2\mathbb{N}$ and the

Algorithm 1: The compact genetic algorithm (cGA) to maximize a function $\mathcal{F} : \{0,1\}^n \to \mathbb{R}$.

```
1  $t \leftarrow 0$;
2  $f_t = (\frac{1}{2}, \dots, \frac{1}{2}) \in [0,1]^n$;
3  repeat
4      $x^1 \leftarrow \text{Sample}(f_t)$;
5      $x^2 \leftarrow \text{Sample}(f_t)$;
6      if $\mathcal{F}(x^1) \geq \mathcal{F}(x^2)$ then $(y^1, y^2) \leftarrow (x^1, x^2)$ else
         $(y^1, y^2) \leftarrow (x^2, x^1)$;
7      $f'_{t+1} \leftarrow f_t + \frac{1}{\mu}(y^1 - y^2)$;
8      $f_{t+1} \leftarrow \text{minmax}(\frac{1}{n}\mathbf{1}_n, f'_{t+1}, (1 - \frac{1}{n})\mathbf{1}_n)$;
9      $t \leftarrow t + 1$;
10 until forever;
```

set of frequencies that can occur is

$$F := F_\mu := \{\tfrac{1}{n} + \tfrac{i}{\mu} \mid i \in [0..n_\mu]\}.$$

This assumption was made, e.g., in the proof of Theorem 2 in [24] and in the paper [23] (see the paragraph following Lemma 2.1).

2.2 Related Work

In all results described in this section, we shall assume that the hypothetical population size is at most polynomial in the problem size n, that is, that there is a constant c such that $\mu \leq n^c$.

The first to conduct a rigorous runtime analysis for the cGA was Droste in his seminal work [14]. He regarded the cGA without frequency boundaries, that is, he just took $f_{t+1} := f'_{t+1}$ in our notation. He showed that this algorithm with $\mu \geq n^{1/2+\varepsilon}$, $\varepsilon > 0$ any positive constant, finds the optimum of the OneMax function defined by

$$\text{OneMax}(x) = \|x\|_1 = \sum_{i=1}^{n} x_i$$

for all $x \in \{0,1\}^n$ with probability at least 1/2 in $O(\mu\sqrt{n})$ iterations [14, Theorem 8].

Droste also showed that this cGA for any objective function $\mathcal{F}$ with unique optimum has an expected runtime of $\Omega(\mu\sqrt{n})$ when conditioning on no premature convergence [14, Theorem 6]. It is easy to see that his proof of the lower bound can be extended to the cGA with frequency boundaries, that is, to Algorithm 1. For this, it suffices to deduce from his drift argument the result that the first time $T_{n/4}$ that the frequency distance $D = \sum_{i=1}^{n}(1 - f_{it})$ is less than $n/4$ satisfies $E[T_{n/4}] \geq \mu\sqrt{n}\frac{\sqrt{2}}{4}$. Since the probability to sample the optimum from a frequency distance of at least $n/4$ is at most

$$\prod_{i=1}^{n} f_{it} = \prod_{i=1}^{n}(1 - (1 - f_{it})) \leq \prod_{i=1}^{n} \exp(-(1 - f_{it}))$$
$$= \exp\left(-\sum_{i=1}^{n}(1 - f_{it})\right) \leq \exp(-n/4),$$

the algorithm with high probability does not find the optimum before time $T_{n/4}$.

Ten years after Droste's work, Sudholt and Witt [24] showed that the $O(\mu\sqrt{n})$ upper bound also holds for the cGA with frequency boundaries. There (but the same should be true for the cGA without boundaries) a hypothetical population size of $\mu = \Omega(\sqrt{n}\log n)$ suffices (recall that Droste required $\mu = \Omega(n^{1/2+\varepsilon})$). The technically biggest progress with respect to upper bounds most likely lies in the fact that the analysis in [24] also holds for the expected optimization time, which means that it also includes the rare case that frequencies reach the lower boundary (see our discussion of the relation of expectations and tail bounds for runtimes of EDAs in Section 2.3). Sudholt and Witt also show that the cGA with frequency boundaries with high probability (and thus also in expectation) needs at least $\Omega(\mu\sqrt{n} + n\log n)$ iterations to optimize OneMax. While the $\mu\sqrt{n}$ lower bound could have been also obtained with methods similar to Droste's (in Lemma 5 we do something very similar), the innocent-looking $\Omega(n\log n)$ bound is surprisingly difficult to prove.

Not much is known for hypothetical population sizes below the order of $\sqrt{n}$. It is clear that then the frequencies will reach the lower boundary of the frequency range, so working with a non-trivial lower boundary like $\frac{1}{n}$ is necessary to prevent premature convergence. The recent lower bound $\Omega(\mu^{1/3}n)$ valid for $\mu = O(\frac{\sqrt{n}}{\log n \log\log n})$ of [23] indicates that already a little below the $\sqrt{n}$ regime significantly larger runtimes occur, but with no upper bounds this regime remains largely not understood.

We refer the reader to the recent survey [21] for more results on the runtime of the cGA on classic unimodal test functions like LeadingOnes and BinVal. Interestingly, nothing was known for non-unimodal functions before the recent work of Hasenöhrl and Sutton [17] on jump functions, which we discussed already in the introduction.

To round off the picture, we briefly describe some typical runtimes of evolutionary algorithms on jump functions. We recall that the n-dimensional jump function with jump size $k \geq 1$ is defined by

$$\text{Jump}_{nk}(x) = \begin{cases} \|x\|_1 + k & \text{if } \|x\|_1 \in [0..n-k] \cup \{n\}, \\ n - \|x\|_1 & \text{if } \|x\|_1 \in [n-k+1\,..\,n-1]. \end{cases}$$

Hence for $k = 1$, we have a fitness landscape isomorphic to the one of OneMax, but for larger values of k there is a fitness valley consisting of the $k-1$ highest sub-optimal fitness levels of the OneMax function. This valley is hard to cross for evolutionary algorithms using standard-bit mutation with mutation rate $\frac{1}{n}$ since with very high probability they need to generate the optimum from one of the local optima, which in a single application of the mutation operator happens only with probability less than n^{-k}. For this reason, e.g., the classic $(\mu+\lambda)$ and (μ,λ) EAs all have a runtime of at least n^k. This was proven formally for the (1 + 1) EA in the classic paper [15], but the argument just given proves the n^k lower bound equally well for all $(\mu+\lambda)$ and (μ,λ) EAs. By using larger mutation rates or a heavy-tailed mutation operator, a $k^{\Theta(k)}$ runtime improvement can be obtained [13], but the runtime remains $\Omega(n^k)$ for k constant.

Asymptotically better runtimes can be achieved when using crossover, though this is harder than expected. The first work in this direction [20], among other results, could show that a simple $(\mu + 1)$ genetic algorithm using uniform crossover with rate $p_c = O(1/kn)$ obtains an $O(\mu n^2 k^3 + 2^{2k} p_c^{-1})$ runtime when the population size is at least $\mu = \Omega(k \log n)$. A shortcoming of this result, already noted by the authors, is that it only applies to uncommonly small crossover rates. Using a different algorithm that first always applies crossover and then mutation, a runtime of $O(n^{k-1} \log n)$ was achieved by Dang et al. [6, Theorem 2]. For $k \geq 3$, the logarithmic factor in the runtime can be removed by using a higher mutation rate. With additional diversity mechanisms, the runtime can be further reduced up to $O(n \log n + 4^k)$, see [5]. In the light of this last result, the insight stemming from the previous work [17] and ours is that the cGA apparently without further modifications supplies the necessary diversity to obtain a runtime of $O(n \log n + 2^{O(k)})$.

Finally, we note that runtimes of $O(n\binom{n}{k})$ and $O(k \log(n)\binom{n}{k})$ were shown for the $(1 + 1)$ IA^{hyp} and the $(1 + 1)$ Fast-IA artificial immune systems, respectively [3, 4].

2.3 Expected Runtimes versus Guarantees with High Probability

We note that our main result as well as the previous one [17] for this problem give runtime bounds that hold with high probability, that is, with probability $1 - o(1)$. However, we do not show a bound on the expected runtime. Let us quickly argue what the differences are, why we chose to prove a high-probability statement, and how to transform EDAs with high-probability guarantees into those with guarantees on the expected runtime. We note that Wegener [25, Section 3] with different arguments also suggests to prefer high-probability guarantees over expected runtimes.

For most evolutionary algorithms a high-probability guarantee can easily be turned into a bound on the expected runtime. If we know that a certain algorithm from any initial state finds the optimum in time T with at least constant probability, then by splitting time into consecutive segments of length T we see that after time γT the probability that the algorithm has not succeeded is at most $\exp(-\Omega(\gamma))$. Consequently, the runtime is stochastically dominated by T times a geometric random variable with constant success rate, and consequently, the expected runtime is $O(T)$. The same argument gives a scalable tail bound of type "with probability at most $\exp(-\Omega(\gamma))$, the runtime is more than γT."

For EDAs, it is usually much harder to show a good performance for any initial situation since there are some states which are particularly unfavorable (usually when all frequencies are close to the wrong boundary value). This does not rule out that the expected runtime and the time that is obtained with high probability are of the same order, but proving the bound on the expected runtime needs stronger arguments. The analysis of the expected runtime of the cGA on OneMax in [24] is an example for such a result.

This additional proof complexity raises the question if this effort is justified if the hardest part is dealing with states of the algorithm that are rarely reached (in [24] with probability $O(n^{-c})$ only, where c can be any positive constant). While we think that is was very valuable that the work [24] showed how to compute expected runtimes for EDAs, we feel that such results are not always needed, both because of the difficulty to obtain such results and because, in some sense, they are a mildly unnatural remedy to the deeper problem.

As said, the main reason why guarantees for the expected runtime of an EDA can be difficult to show is that the EDA with small probability can reach a state from which the optimum is hard to reach. When in such a state, however, instead of spending much time to leave the unfavorable state, it would be more efficient and more natural to simply restart the algorithm and have a new good chance for a fast optimization process. While we cannot expect the algorithm to detect that it is in an unfavorable state (except in the case of premature convergence when no frequency boundaries are used), the following simple parallel-run strategy under mild assumptions can do this automatically. More precisely, via suitable parallel runs we obtain an expected runtime that is only a logarithmic factor above the runtime the EDA would have with high probability when using the optimal population size. Hence this approach both obtains expected runtimes and optimizes the value of the parameter μ.

Parallel EDA runs with exponentially growing population size: Let $\mathcal{A}$ be an EDA with a parameter μ and let $\mathcal{P}$ be a problem we want to solve. We assume that there are unknown values $\tilde{\mu}$ and T such that $\mathcal{A}$ with any parameter value $\mu \geq \tilde{\mu}$ solves $\mathcal{P}$ in time μT with probability at least $\frac{3}{4}$.

We propose the following strategy to solve $\mathcal{P}$ via parallel runs of $\mathcal{A}$ with different parameter values. We start with no process running. In round $i = 1, 2, \ldots$ of our strategy, we let all running processes (which are process 1 to $i - 1$) use a computational budget of 2^{i-1}; further, we start process i with parameter $\mu = \mu_i := 2^{i-1}$ and let it use a budget of $\sum_{j=0}^{i-1} 2^j$. We stop when any process has solved the problem.

We observe that at the end of round i, processes 1 to i are running and have each spent a budget of $\sum_{j=0}^{i-1} 2^j$. Consequently, the total budget spent in the first i rounds is less than $i2^i$.

Note that after round $i_0 = 1 + \lceil \log_2 \tilde{\mu} \rceil + \lfloor \log_2 T \rfloor$, the process with parameter $\mu = 2^{\lceil \log_2 \tilde{\mu} \rceil} \geq \tilde{\mu}$ has started and has used a time budget of

$$\sum_{j=0}^{i_0-1} 2^j \geq \sum_{j=\lceil \log_2 \tilde{\mu} \rceil}^{i_0-1} 2^j = \mu \sum_{j=0}^{\lfloor \log_2 T \rfloor} 2^j \geq \mu T.$$

Consequently, with probability 3/4 this process has found the optimum at that time. With the same type of computation, we see that in round $i_0 + j$, the process with parameter $2^j \mu$ is finished with probability 3/4. Consequently, the round in which we found the solution is dominated by $i_0 - 1$ plus a geometric distribution with success rate 3/4. The expected time taken by this strategy to solve the problem thus is at most

$$\sum_{i=i_0}^{\infty} \left(\frac{1}{4}\right)^{i-i_0} \left(\frac{3}{4}\right) i\, 2^i = \frac{3}{4}\, 2^{i_0} \sum_{j=0}^{\infty} 2^{-j}(j + i_0) = 3 \cdot 2^{i_0-1}(i_0 + 1)$$

using the well-known result $\sum_{j=0}^{\infty} j\, 2^{-j} = 2$. We further estimate the expected runtime of our parallel-run strategy by

$$3 \cdot 2^{i_0-1}(i_0 + 1) \leq 3\mu T(\log_2(\mu T) + 2) \leq 6\tilde{\mu} T(\log_2(\tilde{\mu} T) + 3) =: T_{\text{par}}.$$

We note that, again, analogous arguments give the scalable tail bound that with probability at most $\exp(-\Omega(\gamma))$, the runtime exceeds γT_{par}. We recall here that for EDAs such tail bounds are usually not shown, again due to the fact that the EDA may reach a situation from which is takes a long time to reach the optimum.

We note that if the values of $\tilde{\mu}$ and T were known in advance, then restarting the EDA with $\mu = \tilde{\mu}$ and with a budget of T until the problem is solved would immediately give an algorithm with expected runtime $T^* \leq \frac{4}{3}\tilde{\mu}T$. This is the best-possible expected runtime that can be deduced from our assumptions. Consequently, our parallel-run strategy with its $O(T^* \log T^*)$ expected runtime obtains the optimal expected runtime apart from a logarithmic factor.

We remark that a logarithmic factor usually is not a lot compared to what can be lost by choosing a wrong algorithm parameter, in particular, when the parameter is hard to guess. We note here that the recent work [23] suggests that already for the simple OneMax function, the hypothetical population size has a non-obvious influence on the runtime: Sufficiently small values give an $O(n \log n)$ runtime, in a middle regime the runtime increases to $\tilde{\Omega}(n^{7/6})$ before dropping again to $O(n \log n)$ and then increasing linearly with μ. In the light of such results, a logarithmic overhead for exploiting a near-optimal rate appears to be a good trade-off.

3 MAIN TECHNICAL ANALYSIS

We now conduct our runtime analysis of the cGA on jump functions. We start by giving a rough overview of the proof, then provide the necessary ingredients of the main proof, and finally state and prove our main result.

3.1 Proof Overview

We now give a brief overview of our runtime analysis and show how the different partial results work together. We leave it to the reader to read this section now or after the presentation of the partial results (or twice).

In our analysis, we roughly distinguish three phases of the optimization process. The first phase lasts until for the first time the frequency distance $D_t := n - \|f_t\|_1$ is $O(\log n)$ with a large implicit constant. During this phase, by Lemma 1 and a union bound, with high probability we will never sample a solution in the gap. Consequently, we can pretend that we are optimizing the OneMax function and use our analysis of Lemma 5, which reuses arguments of the classic result by Droste [14] including Lemma 4. The second phase then lasts until we have a D_t value of $O(k)$, again with large implicit constant. In this phase, we use the drift computed in Lemma 6. We profit from the fact that in this phase we only need to obtain a moderate decrease of D_t and apply the additive drift theorem with the smallest drift that can occur in this phase, which is $\Omega(1/\mu)$. Since this phase is so short, a simple Markov bound suffices to show that the phase ends with high probability in due time. Once we reach a D_t value of $O(k)$, we have a reasonable chance to sample the optimum by Lemma 7. Since in this phase samples in the gap occur frequently, we have less control over D_t, in particular, we cannot exhibit an expected decrease of D_t. We therefore pessimistically estimate D_t as if D_t would always increase, which gives (apart from the boundary effects described in Lemma 2) an increase of $|\|x^1\|_1 - \|x^2\|_1|$. Since D_t is small, these increases are small as well, as again ensured by Lemma 1. With this observation, we can argue that we have a D_t value of $O(k)$ for almost μ iterations, which suffices to sample the optimum with high probability.

All the arguments above need that the frequencies are bounded away from the lower boundary of $\frac{1}{n}$, more precisely, that they are $\Omega(1)$ at all times. In the first two phases, we ensure this via Lemma 3, our general result for random processes that are not Markov processes. To this aim, we estimate the probabilities of certain frequency changes by adjusting this data from the OneMax process (Lemma 8, taken from Sudholt and Witt [24]) via a pessimistic estimate of the negative influence of search points sampled in the gap. For the third phase, the fact that this phase only last $o(\mu)$ iterations implies that frequencies change by at most $o(1)$, hence the $\Omega(1)$ lower bound remains intact.

3.2 Technical Ingredients of the Main Proof

In this section, we collect the central arguments needed in the proof of our main result. Since we hope that some arguments are helpful for other runtime analyses of EDAs, we fix no general notation apart from the one defined in Algorithm 1 (at the price of occasionally restating a notion).

We frequently use the following estimate, which states that the OneMax fitness of a search point sampled from Sample(f) is close to the expected OneMax fitness $\|f\|_1$. Since we mostly need such results for frequency vectors close to $(1, \ldots, 1)$, we formulate this result in terms of distances to the maximum values. For reasons of space, the proof of this elementary result is omitted, but can be found in [10].

Lemma 1. *Let* $f \in [0,1]^n$, $D := n - \|f\|_1$, $D^- \leq D \leq D^+$, $x \sim \text{Sample}(f)$, *and* $d(x) := n - \|x\|_1$. *Then for all* $\delta \in [0,1]$, *we have*

$$\Pr[d(x) \geq (1+\delta)D^+] \leq \exp(-\tfrac{1}{3}\delta^2 D^+),$$
$$\Pr[d(x) \leq (1-\delta)D^-] \leq \exp(-\tfrac{1}{2}\delta^2 D^-).$$

We need the lemma above in particular to argue that the probability to sample a search point in the gap region of the Jump function is small. For the Jump$_{nk}$ function, we observe that when $D := n - \|f\|_1$ is at least $2k$, then the probability that $x \sim \text{Sample}(f)$ lies in the gap, that is, has $n-k < \|x\|_1 < n$, is $e^{-\Omega(k)}$. We can also get low constant probabilities for sampling in the gap when $D \geq k + \Omega(\sqrt{k})$ with large implicit constant. In [17, Lemma 3.2], a gap probability of at most $1 - 1/\sqrt{2} \leq 0.293$ is shown when $D \geq k + c$ for c a sufficiently large constant and $k = o(n)$, but we are skeptical that this is true. Note that when $f = \frac{n-k-c}{n}\mathbf{1}_n$, then $X = n - \|x\|_1$ with $x \sim \text{Sample}(f)$ follows a binomial distribution with parameters n and $\frac{k+c}{n}$. Hence if k is large compared to c, then $\Pr[X < k] = \Pr[X < E[X] - c] \approx \frac{1}{2}$.

When, in the notation of Algorithm 1, the current frequency vector f_t is such that $f_{it} \in \{\frac{1}{n}, 1 - \frac{1}{n}\}$ for some $i \in [1..n]$, then it may happen that $f'_{t+1} \notin [\frac{1}{n}, 1 - \frac{1}{n}]$ and consequently f_{t+1} does not satisfy the nice relation $f_{t+1} = f_t + \frac{1}{\mu}(y^1 - y^2)$. The following lemma quantifies these discrepancies by showing that they are stochastically dominated (see, e.g., [7]) by $\frac{1}{\mu}$ times a binomial distributions with expectation 2. The proof is again omitted for reasons of space, but can be found in [10].

Lemma 2. *Let $P = 2\frac{1}{n}(1-\frac{1}{n})$. Let $t \geq 0$. Using the notation given in Algorithm 1, consider iteration $t+1$ of a run of the cGA started with a fixed frequency vector $f_t \in [\frac{1}{n}, 1-\frac{1}{n}]^n$.*

(i) *Let $L := \{i \in [1..n] \mid f_{it} = \frac{1}{n}\}$, $\ell = |L|$, and $M := \{i \in L \mid x_i^1 \neq x_i^2\}$. Then $|M| \sim \text{Bin}(\ell, P)$ and $\|f_{t+1}\|_1 - \|f'_{t+1}\|_1 \leq \|(f_{t+1})_{|L}\|_1 - \|(f'_{t+1})_{|L}\|_1 \leq \frac{1}{\mu}|M| \leq \frac{1}{\mu}\text{Bin}(n, \frac{2}{n})$.*

(ii) *Let $L := \{i \in [1..n] \mid f_{it} = 1-\frac{1}{n}\}$, $\ell = |L|$, and $M := \{i \in L \mid x_i^1 \neq x_i^2\}$. Then $|M| \sim \text{Bin}(\ell, P)$ and $\|f'_{t+1}\|_1 - \|f_{t+1}\|_1 \leq \|(f'_{t+1})_{|L}\|_1 - \|(f_{t+1})_{|L}\|_1 \leq \frac{1}{\mu}|M| \leq \frac{1}{\mu}\text{Bin}(n, \frac{2}{n})$.*

Since sampling the optimum is particularly unlikely when frequencies are close to the lower boundary, we shall argue that the frequencies in a run of the cGA on OneMax stay away from the lower boundary for a decent time.

A similar result was given in [17, Lemma 2.4], however, the proof appears to be not complete. It seems to us that the main technical prerequisite of this result, Lemma 2.2 in [17] with a proof of a little over one page in the condensed proceedings style, is not correct for two reasons. Since the proof of Lemma 2.2 never refers to the frequency boundaries, it is not clear if it is applicable for the cGA with these boundaries. Rather, a frequency vector having one entry $f_{it} = \frac{1}{n}$ and another one $f_{jt} = 1 - \frac{1}{n}$ seems to be a counterexample (note that the frequency vector is called p_t instead of f_t in [17]). However, also for the case without boundaries counterexamples seem to exist for all value of μ, e.g., the frequency vector $f_t = (\frac{1}{100}, \frac{1}{2})$.

We did not see how to repair the otherwise elegant argument via the Azuma-Hoeffding inequality. For this reason, using a sequence of elementary reductions, we argue that the true random process of a frequency, which is not a Markov process when regarding one frequency in isolation, can be pessimistically replaced by a fair random walk on an unbounded frequency domain. For the analysis of the latter, classic Chernoff bounds can be used. This general approach was also taken in [14], however in the easier situation that there are no frequency boundaries and that the objective function is OneMax.

Lemma 3. *Let μ be arbitrary except that it satisfies the well-behaved frequency assumption. Let $\varepsilon > 0$. Let $Z_0, Z_1, \ldots$ be any random process on F_μ such that (i) $Z_0 = \frac{1}{2}$, (ii) for all $t = 0, 1, \ldots$ there are numbers $p_t, q_t, r_t \in [0,1]$, depending on $Z_0, Z_1, \ldots, Z_t$, such that $p_t + q_t + r_t = 1$ and, conditional on $Z_0, \ldots, Z_t$,*

$$\Pr[Z_{t+1} = Z_t] = p_t$$
$$\Pr[Z_{t+1} = Z_t + \tfrac{1}{\mu}] = q_t$$
$$\Pr[Z_{t+1} = Z_t - \tfrac{1}{\mu}] = r_t.$$

We further assume that $r_t = 0$ when $Z_t = \frac{1}{n}$, that $q_t = 0$ if $Z_t = 1-\frac{1}{n}$, and that $q_t \geq r_t$ when $Z_t \neq 1 - \frac{1}{n}$. Then for all $T \in \mathbb{N}$,

$$\Pr[\exists t \in [0..T] : Z_t < \tfrac{1}{2} - \varepsilon] \leq 2\exp\left(-\frac{2\mu^2\varepsilon^2}{T}\right).$$

Proof. We first observe that we can assume $p_t = 0$ for all t. The event $Z_{t+1} = Z_t$ that the process does not move only slows down the process in the sense that it visits fewer states. Similarly, we can assume that $q_t = r_t$ except in the cases $Z_t \in \{\frac{1}{n}, 1-\frac{1}{n}\}$. For this now uniquely defined process, which is an unbiased random walk with reflecting boundaries, we show

$$\Pr[\exists t \in [0..T] : Z_t \notin [\tfrac{1}{2} - \varepsilon, \tfrac{1}{2} + \varepsilon]] \leq 2\exp\left(-\frac{2\mu^2\varepsilon^2}{T}\right).$$

Being interested in the event that the process reaches a state outside $[\frac{1}{2} - \varepsilon, \frac{1}{2} + \varepsilon]$ at least once, we can also drop the boundary conditions and assume that we have $Z_{t+1} \in \{Z_t - \frac{1}{\mu}, Z_t + \frac{1}{\mu}\}$ uniformly at random at all times t. We can now rewrite the Z_t as follows. Let $X_1, \ldots, X_T$ be independent random variables uniformly distributed on $\{-\frac{1}{\mu}, \frac{1}{\mu}\}$. Then for all t, Z_t has the same distribution as $\frac{1}{2} + \sum_{i=1}^t X_t$. Consequently, by the additive Chernoff bound (in the sharper version working also for maxima, see (2.17) and Theorem 2 in [19] or, e.g., Theorem 10.31 together with Theorem 10.9 in [8]), we have

$$\begin{aligned}\Pr[\exists t \in [0..T] &: Z_t \notin [\tfrac{1}{2} - \varepsilon, \tfrac{1}{2} + \varepsilon]] \\ &= \Pr[\exists t \in [0..T] : |Z_t - E[Z_t]| > \varepsilon] \\ &\leq 2\exp\left(-\frac{2\varepsilon^2}{T(1/\mu)^2}\right) = 2\exp\left(-\frac{2\mu^2\varepsilon^2}{T}\right).\end{aligned}$$

□

The following result is a weaker form of what was shown in the proof of Lemma 5 in [14].

Lemma 4. *There is a constant $C > 0$ such that the following holds. Let $n \in \mathbb{N}$ and $D \in \mathbb{N}$. Let $f \in [\frac{1}{3}, 1]^n$ such that $\|f\|_1 \leq n - D$. Let $x^1, x^2 \sim \text{Sample}(f)$ independently. Then*

$$\Pr\left[\left|\|x^1\|_1 - \|x^2\|_1\right| \geq \tfrac{1}{5}\sqrt{D}\right] \geq C.$$

Since we shall use that the optimization process of the cGA on a jump function is identical to the one on the OneMax function as long as no search point in the gap region is sampled, we find the following analysis of the optimization process on OneMax useful. It differs from Droste's analysis of the cGA on OneMax [14] in that it regards the cGA with boundaries and in that it proves a high-probability statement for reaching a near-optimal frequency vector. We note that our analysis can easily be extended to also give a bound for the time to sample an optimal solution, but we do not need such a result (and in fact, such a result is implied by our main result). Also, a simplified version of our proof would apply to the cGA without boundaries.

Lemma 5. *Consider a run of the cGA with $\mu \geq \log_2 n$ on the OneMax benchmark function. Let $D_t := n - \|f_t\|_1$ for all t. Let K be a sufficiently large constant. Let T be the first time that $D_t \leq K$ or that there is an $i \in [1..n]$ with $f_{it} < \frac{1}{3}$. Then*

$$\Pr\left[T \geq \frac{10(2+\sqrt{2})}{C}\mu\sqrt{n}\right] = \exp(-\Omega(\mu)),$$

where C is the constant from Lemma 4.

For reasons of space, the formal proof of this lemma is omitted in this extended abstract. It can be found in the preprint [10].

We now analyze the drift in D_t when we are that close to the gap that we cannot assume anymore that we never sample a search point in the gap. To be precise, let us define the gap by $G := G_{nk} := \{x \in \{0,1\}^n \mid n - k < \|x\|_1 < n\}$. Let $G^+ := G \cup \{(1, \ldots, 1)\}$.

A difficulty here, which was not treated fully rigorously in [17, Lemma 3.1], is that the event G_t that x^1 or x^2 lie in the gap and the random variable $|\|x^1\|_1 - \|x^2\|_1|$ are not independent. Consequently, the estimate $E[D_t - D_{t+1} \mid D_t] = \frac{1}{\mu}|\|x^1\|_1 - \|x^2\|_1|(1 - 2\Pr[G_t])$ is not correct. In fact, the correlation is indeed not in our favor. When $|\|x^1\|_1 - \|x^2\|_1|$ is large, the probability that a search point in the gap was sampled (and thus the frequency update is done in the unwanted direction) is higher.

Lemma 6. *Let μ be arbitrary satisfying the well-behaved frequency assumption. Let $k \in [1..\frac{1}{2}n - 1]$. Consider an iteration t of the cGA optimizing* Jump$_{nk}$ *started with a frequency vector f_t such that $D_t = n - \|f_t\|_1 \geq 2k$ and such that $f_{it} \geq \frac{1}{3}$ for all $i \in [1..n]$. Then*

$$E[\mu D_t - \mu D_{t+1}] \geq \tfrac{1}{5}C\sqrt{D_t} - 6D_t \exp(-\tfrac{1}{8}D_t) - 2,$$

where C is the constant from Lemma 4.

Proof. From the definition of the cGA, we note that when x^1 and x^2 are both not in G^+, then $D'_{t+1} := n - \|f'_{t+1}\|_1$ satisfies $D'_{t+1} = D_t - \frac{1}{\mu}|\|x^1\|_1 - \|x^2\|_1|$. In all other cases, we have $D'_{t+1} \leq D_t + \frac{1}{\mu}|\|x^1\|_1 - \|x^2\|_1|$. Consequently,

$$\begin{aligned}
&E[\mu D_t - \mu D'_{t+1}]\\
&\geq \Pr[x^1, x^2 \notin G^+]\, E[|\|x^1\|_1 - \|x^2\|_1| \mid x^1, x^2 \notin G^+]\\
&\quad - \Pr[\{x^1, x^2\} \cap G^+ \neq \emptyset]\, E[|\|x^1\|_1 - \|x^2\|_1| \mid \{x^1, x^2\} \cap G^+ \neq \emptyset]\\
&= E[|\|x^1\|_1 - \|x^2\|_1|]\\
&\quad - 2\Pr[\{x^1, x^2\} \cap G^+ \neq \emptyset]\, E[|\|x^1\|_1 - \|x^2\|_1| \mid \{x^1, x^2\} \cap G^+ \neq \emptyset].
\end{aligned}$$

When the frequencies are all at least $\frac{1}{3}$, we conclude from Lemma 4 that $E[|\|x^1\|_1 - \|x^2\|_1|] \geq \frac{1}{5}C\sqrt{D_t}$.

For the contribution when search points are in G^+, we first note that the second bound of Lemma 1 (with $\delta = \frac{1}{2}$ and $D^- = D_t$) and $D_t \geq 2k$ yield

$$\Pr[x^1 \in G^+] \leq \Pr[d(x^1) \leq \tfrac{1}{2}D_t] \leq \exp(-\tfrac{1}{8}D_t).$$

Then, exploiting the symmetry between x^1 and x^2, counting the case $x^1, x^2 \in G^+$ twice, and using again $\frac{1}{2}D_t \geq k$, we compute

$$\begin{aligned}
&\Pr[\{x^1, x^2\} \cap G^+ \neq \emptyset]\, E[|\|x^1\|_1 - \|x^2\|_1| \mid \{x^1, x^2\} \cap G^+ \neq \emptyset]\\
&\quad \leq 2\Pr[x^1 \in G^+]\, E[|\|x^1\|_1 - \|x^2\|_1| \mid x^1 \in G^+]\\
&\quad \leq 2\Pr[x^1 \in G^+]\, \Big(E[|\|x^1\|_1 - n| \mid x^1 \in G^+] + E[|n - \|x^2\|_1|]\Big)\\
&\quad \leq 2\Pr[x^1 \in G^+]\, (k + D_t)\\
&\quad \leq 2\exp(-\tfrac{1}{8}D_t)(\tfrac{1}{2}D_t + D_t) = 3\exp(-\tfrac{1}{8}D_t)D_t.
\end{aligned}$$

In summary, we have

$$E[\mu D_t - \mu D'_{t+1}] \geq \tfrac{1}{5}C\sqrt{D_t} - 6D_t \exp(-\tfrac{1}{8}D_t).$$

By Lemma 2, we further have $E[\mu D_{t+1} - \mu D'_{t+1}] \leq 2$. Consequently, recalling that the linearity of expectation holds also for dependent random variables, we have $E[\mu D_t - \mu D_{t+1}] = E[\mu D_t - \mu D'_{t+1}] - E[\mu D_{t+1} - \mu D'_{t+1}] \geq \frac{1}{5}C\sqrt{D_t} - 6D_t \exp(-\frac{1}{8}D_t) - 2$. □

The following elementary estimate gives a lower bound for the probability to sample the optimum. For reasons of space, its proof can only be found in the preprint [10].

Lemma 7. *Let $0 < c < 1$ and $f \in [c, 1]^n$. Let $x \sim$ Sample(f). Then $\Pr[x = (1, \dots, 1)] \geq c^{(n - \|f\|_1)/(1-c)}$.*

We shall use the following estimate on the expected change of a frequency that is not affected by the boundaries. This result was proven in [24, Lemma 3].

Lemma 8. *Let μ be arbitrary. Consider a run of the cGA optimizing* OneMax. *Consider an iteration starting with a frequency vector f_t. Let $i \in [1..n]$ be such that $\frac{1}{n} + \frac{1}{\mu} \leq f_{it} \leq (1 - \frac{1}{n}) - \frac{1}{\mu}$. Then*

$$E[f_{i,t+1} - f_{it}] \geq \frac{2}{11}\frac{f_{it}(1 - f_{it})}{\mu}\left(\sum_{j \neq i} f_{jt}(1 - f_{jt})\right)^{-1/2}.$$

3.3 Main Result and Proof

We are now ready to state and prove our main result.

Theorem 9. *Let $k \leq \frac{1}{20}\ln(n) - 1$. Let $\mu \geq K\sqrt{n}\ln(n)$ for a sufficiently large constant K, but polynomially bounded in n. Then the cGA with frequency boundaries (Algorithm 1) with hypothetical population size μ with probability $1 - o(1)$ finds the optimum of the* Jump$_{nk}$ *function in time $O(n \log n)$.*

Proof. To allow the reader to easily check that all implicit constants can be chosen in a way that they give the claimed results, we make these constants explicit in the following proof, but note that for most of them it just suffices to choose them sufficiently large.

Let $k \leq C_k \ln(n) - 1$, $C_k = \frac{1}{20}$. Let $\mu \geq c_\mu\sqrt{n}\ln(n)$ and $\mu \leq n^{C_\mu}$ for a constant c_μ to be defined in a moment and, say, $C_\mu \geq 1$. Consider a run of the cGA on the objective function Jump$_{nk}$.

Let $\tilde{T}'$ be the first time that $D_t := n - \|f_t\|_1$ satisfies $D_t \leq D' := C_{D'}\ln n$, where $C_{D'} \geq 8C_\mu + 12$ is a constant. Let $\tilde{T}''$ be the first time that $D_t \leq D'' := \max\{2k+1, C_{D''}\}$, where $C_{D''}$ is a sufficiently large constant (independent also of all other constants).

Let $T' = \min\{\tilde{T}', \lfloor C_{T'}\mu\sqrt{n}\rfloor\}$ with $C_{T'} = \frac{10(2+\sqrt{2})}{C}$, where C is the constant from Lemma 4. Let $T'' = \min\{\tilde{T}'', \lfloor C_{T''}\mu\sqrt{n}\rfloor\}$ with $C_{T''} = C_{T'} + 1$. Let now $c_\mu \geq 36C_{T''}$.

We first argue that with high probability we have no frequencies below $\frac{1}{3}$ up to time T''. For this, consider some time t such that $f_t \in [\frac{1}{3}, 1]^n$ and $D_t \geq D''$. Consider a fixed bit $i \in [1..n]$ such that $f_{it} \neq 1 - \frac{1}{n}$. If we were optimizing the OneMax function, then by Lemma 8,

$$\begin{aligned}
&\Pr[f_{i,t+1} = f_{it} + \tfrac{1}{\mu}] - \Pr[f_{i,t+1} = f_{it} - \tfrac{1}{\mu}]\\
&\quad = \mu E[f_{i,t+1} - f_{it}]\\
&\quad \geq \frac{2}{11} f_{it}(1 - f_{it})\left(\sum_{j \neq i} f_{jt}(1 - f_{jt})\right)^{-1/2}\\
&\quad \geq \frac{2}{11} f_{it}(1 - f_{it})\,(D_t)^{-1/2}.
\end{aligned}$$

Regardless of whether we optimize OneMax or Jump$_{nk}$, the events $f_{i,t+1} = f_{it} + \frac{1}{\mu}$ and $f_{i,t+1} = f_{it} - \frac{1}{\mu}$ can only occur when the two search points sampled in this iteration satisfy $x^1_i \neq x^2_i$. Further, the definition of $f_{i,t+1}$ differs from the OneMax case at most when at least one of x^1 and x^2 lie in the gap G_{nk}. Hence the following coarse correction of the above estimate is valid for the optimization of $\textsc{Jump}_{nk}$.

$$\Pr[f_{i,t+1} = f_{it} + \tfrac{1}{\mu}] - \Pr[f_{i,t+1} = f_{it} - \tfrac{1}{\mu}]$$
$$\geq \tfrac{2}{11} f_{it}(1-f_{it})(D_t)^{-1/2} - \Pr[(x_i^1 \neq x_i^2) \wedge (\{x^1,x^2\} \cap G_{nk} \neq \emptyset)].$$

We now estimate this correction term, first by noting that $\Pr[(x_i^1 \neq x_i^2) \wedge (\{x^1,x^2\} \cap G_{nk} \neq \emptyset)] = \Pr[x_i^1 \neq x_i^2] \cdot \Pr[\{x^1,x^2\} \cap G_{nk} \neq \emptyset \mid x_i^1 \neq x_i^2]$, then by using the union bound estimate $\Pr[\{x^1,x^2\} \cap G_{nk} \neq \emptyset \mid x_i^1 \neq x_i^2] \leq 2\Pr[x^1 \in G_{nk} \mid x_i^1 \neq x_i^2]$. Conditional on $x_i^1 \neq x_i^2$, the bit string x^1 is sampled from $\text{Sample}(f_t)$, however, conditional on the i-th bit being zero or one. In either case, to have $x^1 \in G_{nk}$, we need that $\tilde{D} = \sum_{j \neq i}(1 - x_j^1)$ is at most $k \leq \frac{1}{2}(D_t - 1)$, where we recall that $D_t \geq D'' \geq 2k+1$. Since $E[\tilde{D}] = D_t - (1 - f_{it}) \geq D_t - 1$, by Lemma 1 with $\delta = \frac{1}{2}$ this event happens with probability at most $\exp(-\frac{1}{8}(D_t - 1))$. Together with $\Pr[x_i^1 \neq x_i^2] = 2f_{it}(1 - f_{it})$, we obtain

$$\Pr[f_{i,t+1} = f_{it} + \tfrac{1}{\mu}] - \Pr[f_{i,t+1} = f_{it} - \tfrac{1}{\mu}]$$
$$\geq \tfrac{2}{11} f_{it}(1-f_{it})(D_t)^{-1/2} - 2f_{it}(1-f_{it})\exp(-\tfrac{1}{8}(D_t - 1)),$$

which is non-negative since $D_t \geq D'' \geq C_{D''}$, which was chosen sufficiently large.

Consequently, the process $(f_{it})_t$ satisfies the assumptions of Lemma 3 up to time T''. If $T'' < C_{T''}\mu\sqrt{n}$, we artificially extend the process (for the following argument only) by setting $f_{it} = f_{iT''}$ for all $t \in [T'' + 1..C_{T''}\mu\sqrt{n}]$. By Lemma 3 we thus obtain that up to time $T = \lfloor C_{T''}\mu\sqrt{n} \rfloor$, the i-th frequency is always at least $\frac{1}{3}$ with probability $1 - 2\exp(-\frac{\mu^2}{18T}) \geq 1 - 2\exp(-\frac{\mu}{18C_{T''}\sqrt{n}}) \geq 1 - 2\exp(-\frac{c_\mu}{18C_{T''}} \ln n)$. With a union bound over the n frequencies, we have $f_t \in [\frac{1}{3}, 1]^n$ in this time interval with probability at least $1 - 2n\exp(-\frac{c_\mu}{18C_{T''}} \ln n) = 1 - O(1/n)$ by choice of c_μ and $C_{T''}$.

Since $D' = C_{D'} \ln n$ with $C_{D'} \geq 12$ and $k \leq C_k \ln n \leq \frac{C_{D'}}{2} \ln n$, by Lemma 1 and a union bound the probability that within the first $T' \leq C_{T'}\mu\sqrt{n}$ iterations a search point in the gap region is sampled, is at most $2C_{T'}\mu\sqrt{n}\exp(-\frac{C_{D'}}{8} \ln n) \leq 2C_{T'} n^{C_\mu + 0.5 - C_{D'}/8} = O(1/n)$. Consequently, by Lemma 5, after at most $C_{T'}\mu\sqrt{n}$ iterations with probability $1 - O(1/n)$ we have $D_t \leq D'$.

We now estimate the additional time it takes to reach $D_t \leq D''$. Let t_0 be the first time such that $D_t \leq D'$. By Lemma 6 and using our assumption that $C_{D''}$ is a large absolute constant, we have $E[D_t - D_{t+1} \mid D_t] \geq \frac{1}{\mu}$ when $D_t \geq D'' \geq C_{D''}$. We define a random process $\tilde{D}_t$ as follows. Let $t \geq t_0$. If $D_s < D''$ for some $s \in [t_0..t]$, then $\tilde{D}_{t-t_0} = 0$. Otherwise, $\tilde{D}_{t-t_0} = D_t$. By the above observation, we have $E[\tilde{D}_t - \tilde{D}_{t+1} \mid \tilde{D}_t > 0] \geq \frac{1}{\mu}$. By the additive drift theorem [18], also to be found in the recent survey [22], $T := \min\{t \mid \tilde{D}_t = 0\}$ satisfies $E[T] \leq \frac{D'}{1/\mu} = O(\mu \log n)$. By Markov's inequality, we have $T = O(\mu n^{0.4} \log n)$ with probability $1 - n^{-0.4}$.

Let now t_0 be such that $D_{t_0} \leq D''$ and all frequencies are at least $\frac{1}{3}$. We first argue that if $D_t \leq \ln(n)^2$, then $\Pr[D_{t+1} \geq D_t + \frac{4}{\mu}\ln(n)^2] \leq n^{\omega(1)}$. By Lemma 1, we have $\Pr[d(x^j) \geq 2\ln(n)^2] \leq \exp(-\ln(n)^2/3) = n^{\omega(1)}$ for $j = 1, 2$. Consequently, with probability $1 - n^{\omega(1)}$, we have both $\|x^1\|_1 \geq n - 2\ln(n)^2$ and $\|x^2\|_1 \geq n - 2\ln(n)^2$. In this case, the Hamming distance between x^1 and x^2 satisfies $H(x^1, x^2) \leq 4\ln(n)^2$, which implies that $|D_t - D_{t+1}| \leq \|f_t - f_{t+1}\|_1 \leq \frac{4}{\mu}\ln(n)^2$ and thus $D_{t+1} \leq D_t + \frac{4}{\mu}\ln(n)^2$. By a union bound, with probability $1 - n^{\omega(1)}$, this happens in all iterations $t_0, \ldots, t_0 + \lfloor \frac{D''}{(4/\mu)\ln(n)^2} \rfloor - 1$ and consequently, throughout these $L = \lfloor \frac{D''}{(4/\mu)\ln(n)^2} \rfloor$ iterations we have $D_t \leq 2D''$. Note that $L = O(\mu/\log(n))$, hence throughout this period we also have $f_{it} \geq \frac{1}{3} - \frac{1}{\mu}L \geq 0.32$ (assuming n to be sufficiently large). By Lemma 7, the probability that a fixed search point sampled in this period is the optimum, is at least $0.32^{2D''/0.68} \geq 0.32^{4C_k \ln(n)/0.68} \geq n^{-6.71C_k} \geq n^{-0.34}$ by choice of C_k. Hence the probability that the optimum is not sampled in this period is at most $(1 - n^{-0.34})^{2L} \leq \exp(-n^{-0.34} \cdot \mu/\ln(n)^2) \leq \exp(-\Omega(n^{0.16}/\log(n)))$. □

Let us remark that we did not try to optimize the implicit constants, nor did we try to find the largest constant C_k such that the $O(n \log n)$ runtime guarantee holds for all $k \leq C_k \ln(n) - 1$. We further note that all but one argument in the above proof, by choosing the constants right, would give a success probability of $1 - n^{-c}$, where c can be any constant. This is not true for the Markov bound argument in the analysis of the time to reach a D_t value of at most D''. Without further details, we note that also for this phase an arbitrary inverse-polynomial failure probability could be obtained with stronger methods. Finally, we note that by taking $k = 1$, our result also applies to the OneMax function.

4 CONCLUSION

This is, to the best of our knowledge, only the second mathematical analysis of an EDA on a multi-modal optimization problem. Our main result shows that the cGA can optimize jump functions with logarithmic jump sizes in asymptotically the same efficiency as the simple OneMax function. It thus does not suffer from the fitness valleys present in these objective function.

The obvious question arising from this work is to what extent such or similar results hold for other EDAs. Natural candidates could be the UMDA, for which several rigorous runtime results exist, see [21], and the significance-based cGA [12], which might profit from using only the three frequencies $\frac{1}{n}$, $\frac{1}{2}$, and $1 - \frac{1}{n}$. Equally interesting would be results for other optimization problems. We would speculate that our result easily extends to the plateau function defined in [1], so more interesting would be combinatorial optimization problems with significant local optima or plateaus.

On the more speculative side, given that the black-box complexity of jump functions is low even for large jump sizes [2, 11], one could also try to challenge the upper bound $\exp(O(k))$ given in [17] for larger values of k. Since [9] shows a matching lower bound of $\exp(\Omega(k))$, the only way to improve the exponential dependence on k would be via a modification of the cGA. We have no idea to what extent this is possible, but given that [17] and this work show that EDAs can be more powerful on multi-modal functions, this is an interesting direction for future research.

ACKNOWLEDGMENTS

This work was supported by a public grant as part of the Investissement d'avenir project, reference ANR-11-LABX-0056-LMH, LabEx LMH, in a joint call with the Gaspard Monge Program for optimization, operations research and their interactions with data sciences.

REFERENCES

[1] Denis Antipov and Benjamin Doerr. 2018. Precise runtime analysis for plateaus. In *Parallel Problem Solving From Nature, PPSN 2018*. Springer, 117–128.

[2] Maxim Buzdalov, Benjamin Doerr, and Mikhail Kever. 2016. The unrestricted black-box complexity of jump functions. *Evolutionary Computation* 24 (2016), 719–744.

[3] Dogan Corus, Pietro Simone Oliveto, and Donya Yazdani. 2017. On the runtime analysis of the Opt-IA artificial immune system. In *Genetic and Evolutionary Computation Conference, GECCO 2017*. ACM, 83–90.

[4] Dogan Corus, Pietro Simone Oliveto, and Donya Yazdani. 2018. Fast artificial immune systems. In *Parallel Problem Solving from Nature, PPSN 2018*. Springer, 67–78.

[5] Duc-Cuong Dang, Tobias Friedrich, Timo Kötzing, Martin S. Krejca, Per Kristian Lehre, Pietro Simone Oliveto, Dirk Sudholt, and Andrew M. Sutton. 2016. Escaping local optima with diversity mechanisms and crossover. In *Genetic and Evolutionary Computation Conference, GECCO 2016*. ACM, 645–652.

[6] Duc-Cuong Dang, Tobias Friedrich, Timo Kötzing, Martin S. Krejca, Per Kristian Lehre, Pietro Simone Oliveto, Dirk Sudholt, and Andrew M. Sutton. 2018. Escaping local optima using crossover with emergent diversity. *IEEE Transactions on Evolutionary Computation* 22 (2018), 484–497.

[7] Benjamin Doerr. 2018. Better runtime guarantees via stochastic domination. In *Evolutionary Computation in Combinatorial Optimization, EvoCOP 2018*. Springer, 1–17.

[8] Benjamin Doerr. 2018. Probabilistic Tools for the Analysis of Randomized Optimization Heuristics. *CoRR* abs/1801.06733 (2018). arXiv:1801.06733

[9] Benjamin Doerr. 2019. An Exponential Lower Bound for the Runtime of the cGA on Jump Functions. *CoRR* abs/xxx (2019). arXiv:xxx

[10] Benjamin Doerr. 2019. A Tight Runtime Analysis for the cGA on Jump Functions—EDAs Can Cross Fitness Valleys at No Extra Cost. *CoRR* abs/1903.10983 (2019). arXiv:1903.10983

[11] Benjamin Doerr, Carola Doerr, and Timo Kötzing. 2015. Unbiased black-box complexities of jump functions. *Evolutionary Computation* 23 (2015), 641–670.

[12] Benjamin Doerr and Martin S. Krejca. 2018. Significance-based estimation-of-distribution algorithms. In *Genetic and Evolutionary Computation Conference, GECCO 2018*. ACM, 1483–1490.

[13] Benjamin Doerr, Huu Phuoc Le, Régis Makhmara, and Ta Duy Nguyen. 2017. Fast genetic algorithms. In *Genetic and Evolutionary Computation Conference, GECCO 2017*. ACM, 777–784.

[14] Stefan Droste. 2006. A rigorous analysis of the compact genetic algorithm for linear functions. *Natural Computing* 5 (2006), 257–283.

[15] Stefan Droste, Thomas Jansen, and Ingo Wegener. 2002. On the analysis of the (1+1) evolutionary algorithm. *Theoretical Computer Science* 276 (2002), 51–81.

[16] Georges R. Harik, Fernando G. Lobo, and David E. Goldberg. 1999. The compact genetic algorithm. *IEEE Transactions on Evolutionary Computation* 3 (1999), 287–297.

[17] Václav Hasenöhrl and Andrew M. Sutton. 2018. On the runtime dynamics of the compact genetic algorithm on jump functions. In *Genetic and Evolutionary Computation Conference, GECCO 2018*. ACM, 967–974.

[18] Jun He and Xin Yao. 2001. Drift analysis and average time complexity of evolutionary algorithms. *Artificial Intelligence* 127 (2001), 51–81.

[19] Wassily Hoeffding. 1963. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association* 58 (1963), 13–30.

[20] Thomas Jansen and Ingo Wegener. 2002. The analysis of evolutionary algorithms—a proof that crossover really can help. *Algorithmica* 34 (2002), 47–66.

[21] Martin S. Krejca and Carsten Witt. 2018. Theory of Estimation-of-Distribution Algorithms. *CoRR* abs/1806.05392 (2018). arXiv:1806.05392

[22] Johannes Lengler. 2017. Drift Analysis. *CoRR* abs/1712.00964 (2017). arXiv:1712.00964

[23] Johannes Lengler, Dirk Sudholt, and Carsten Witt. 2018. Medium step sizes are harmful for the compact genetic algorithm. In *Genetic and Evolutionary Computation Conference, GECCO 2018*. ACM, 1499–1506.

[24] Dirk Sudholt and Carsten Witt. 2016. Update strength in EDAs and ACO: How to avoid genetic drift. In *Genetic and Evolutionary Computation Conference, GECCO 2016*. ACM, 61–68.

[25] Ingo Wegener. 2005. Simulated annealing beats Metropolis in combinatorial optimization. In *Automata, Languages and Programming, ICALP 2005*. Springer, 589–601.

[26] Weijie Zheng, Guangwen Yang, and Benjamin Doerr. 2018. Working principles of binary differential evolution. In *Genetic and Evolutionary Computation Conference, GECCO 2018*. ACM, 1103–1110.