Choosing the Right Algorithm With Hints From Complexity Theory

Shouda Wang, Weijie Zheng, Benjamin Doerr

Choosing the Right Algorithm With Hints From Complexity Theory*

Shouda Wang
Laboratoire d'Informatique (LIX)
École Polytechnique, CNRS
Institut Polytechnique de Paris
Palaiseau, France

Weijie Zheng
Department of Computer Science and Engineering
Southern University of Science and Technology
Shenzhen, China

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

Abstract

Choosing a suitable algorithm from the myriads of different search heuristics is difficult when faced with a novel optimization problem. In this work, we argue that the purely academic question of what could be the best possible algorithm in a certain broad class of black-box optimizers can give fruitful indications in which direction to search for good established optimization heuristics. We demonstrate this approach on the recently proposed DLB benchmark, for which the only known results are $O(n^3)$ runtimes for several classic evolutionary algorithms and an $O(n^2 \log n)$ runtime for an estimation-of-distribution

*Extended version of the 7-page conference paper in the proceedings of IJCAI 2021 [WZD21]. Corresponding authors: Weijie Zheng and Benjamin Doerr.

algorithm. Our finding that the unary unbiased black-box complexity is only $O(n^2)$ suggests the Metropolis algorithm as an interesting candidate and we prove that it solves the DLB problem in quadratic time. Since we also prove that better runtimes cannot be obtained in the class of unary unbiased algorithms, we shift our attention to algorithms that use the information of more parents to generate new solutions. An artificial algorithm of this type having an $O(n \log n)$ runtime leads to the result that the significance-based compact genetic algorithm (sig-cGA) can solve the DLB problem also in time $O(n \log n)$. Our experiments show a remarkably good performance of the Metropolis algorithm, clearly the best of all algorithms regarded for reasonable problem sizes.

1 Introduction

Randomized search heuristics such as hillclimbers, evolutionary algorithms, ant colony optimizers, or estimation-of-distribution algorithms (EDAs) have been very successful in solving optimization problems for which no established problem-specific algorithm exists. As such, they are applied massively to novel problems for which some understanding of the problem and the desired solution exists, but little algorithmic expertise.

When faced with a novel optimization problem, one has the choice between a large number of established heuristics (and an even larger number of recent metaphor-based heuristics [Sör15]). Which of them to use is a difficult question. Since implementing a heuristic and adjusting it to the problem to be solved can be very time-consuming, ideally one does not want to experiment with too many different heuristics. For that reason, a theory-guided prior suggestion could be very helpful. This is what we aim at in this work. We note that the theory of randomized search heuristics has helped to understand these algorithms, has given suggestions for parameter settings, and has even proposed new operators and algorithms (see the textbooks [NW10, AD11, Jan13, DN20] or the tutorial [Doe20b]), but we are not aware of direct attempts to aid the initial choice of the basic algorithm to be used (as with experimental work, there always is the indirect approach to study the existing results and try to distill from them some general rule which algorithms perform well on which problems, but in particular for the theory domain it is not clear how effective this approach is at the moment).

What we propose in this work is a heuristic approach building on the notion of black-box complexity, first introduced by Droste, Jansen, Tinnefeld, and Wegener [DJTW02] (journal version [DJW06]). In very simple words, the (unrestricted) black-box complexity of an optimization problem is the

performance of the best black-box optimizer for this problem. It is thus a notion not referring to a particular class of search heuristics such as genetic algorithms or EDAs. Black-box complexity has been used successfully to obtain universal lower bounds. Knowing that the black-box complexity of the Needle problem is exponential [DJW06], we immediately know that no genetic algorithm, ant colony optimizer, or EDA can solve the Needle problem in subexponential time. With specialized notions of black-box complexity, more specific lower bounds can be obtained. The result that the unary unbiased black-box complexity of the OneMax benchmark is at least of the order $n \log n$ [LW12] implies that many standard mutation-based evolutionary algorithms cannot optimize OneMax faster than this bound.

With a more positive perspective, black-box complexity has been used invent new algorithms. Noting that the unary unbiased black-box complexity of OneMax is $\Omega(n \log n)$, but the two-ary (i.e., allowing variation operators taking two parents as input) unbiased black-box complexity is only $O(n)$ [DJK+11], a novel crossover-based evolutionary algorithm was developed in [DDE15]. Building on the result that the λ-parallel black-box complexity of the OneMax problem is only $O(\frac{n\lambda}{\log \lambda} + n \log n)$ [BLS14, LS20], dynamic, self-adjusting, and self-adapting EAs obtaining this runtime have been constructed [BLS14, DGWY19, DWY21].

In this work, we also aim at profiting from the guidance of black-box results, however not by designing new algorithms, but by giving an indication which of the existing algorithms could be useful for a particular problem. Compared to the approach taken in [DDE15], we can thus profit from the numerous established and well-understood algorithms and avoid the risky and time-consuming road of developing a new algorithm.

In simple words, what we propose is trying to find out which classes of black-box algorithms contain fast algorithms for the problem at hand. These algorithms may well be artificial as we use them only to determine the direction in which to search for a good established algorithm for our problem. Only once we are sufficiently optimistic that a certain property of black-box algorithms is helpful for our problem, we regard the established heuristics in this class and see if one of them indeed has a good performance.

To show that this abstract heuristic approach towards selecting a good algorithm can indeed be successful, we regard the DeceivingLeadingBlocks (DLB) problem recently proposed by Lehre and Nguyen [LN19]. Lehre and Nguygen conducted rigorous runtime analyses of several classic evolutionary algorithms, all leading to runtime guarantees of order $O(n^3)$ with optimal parameter choices. For the EDA *univariate marginal distribution algorithm (UMDA)*, a runtime guarantee of $O(n^2 \log n)$ was proven in [DK20b]. No other proven upper bounds on runtimes of randomized search heuristics on

the DLB problem existed prior to this work. With only these results from only two prior works, it is safe to call the DLB problem relatively novel and thus an interesting object for our investigation.

Finding more efficient randomized search heuristics: We note that the classic algorithms regarded in [LN19] are all elitist evolutionary algorithms or non-elitist algorithms with parameter settings that let them imitate an elitist behavior. This choice was natural given the weaker understanding of non-elitist algorithms and the fact that not many theoretical works could show a convincing advantage of non-elitist algorithms (see Section 3.4.2). To obtain a first indication whether it is worth investigating non-elitist algorithms for this problem, we show two results. (i) We prove that the $(1+1)$ elitist unbiased black-box complexity of the DLB problem is $\Omega(n^3)$. (ii) We show that there is a simple, artificial, $(1+1)$-type non-elitist unbiased black-box algorithm solving the DLB problem in quadratic time. These two findings motivate us to analyze the existing $(1+1)$-type non-elitist heuristics. Among them, we find that the Metropolis algorithm [MRR$^+$53] with a suitable temperature also optimizes DLB in time $O(n^2)$. We note that there are very few runtime analyses on the Metropolis algorithm (see Section 3.4.1), so it is clear that a synopsis of the existing runtime analysis literature would not have easily suggested this algorithm.

To direct our search for possible further runtime improvements, we first show that the unary unbiased black-box complexity of DLB is at least quadratic. Consequently, if we want to stay in the realm of unbiased algorithms (which we do) and improve beyond quadratic runtimes, then we necessarily have to regard algorithms that are not unary, that is, that generate offspring using the information from at least two parents. That this is possible, at least in principle, follows from our next result, which is an artificial crossover-based algorithm that solves DLB in time $O(n \log n)$. While, together with the previously shown lower bound, it is clear that this performance relies on the fact that offspring are generated from the information of more than one parent, the working principles of this algorithm also include a learning aspect. The algorithm optimizes the blocks of the DLB problem in a left-to-right fashion, but once a block is optimized, it is never touched again. Such learning mechanisms are rarely found in standard evolutionary algorithms, but are the heart of EDAs with their main goal of learning a distribution that allows sampling good solutions. Note that the distribution in an EDA carries information from many previously generated solutions, hence EDAs necessarily generate new solutions based on the information of many parents. For these reasons, we focus on EDAs. We do not find a classic EDA for which we can prove that it solves the DLB problem in subquadratic

time, but we succeed for the significance-based EDA [DK20a] and we show that it optimizes the DLB problem in an expected runtime of $O(n \log n)$.

Overall, these results demonstrate that our heuristic theory-guided approach towards selecting good algorithms for a novel problem can indeed be helpful. We note in particular that the previous works on the DLB problem have not detected that the Metropolis algorithm is an interesting candidate for solving this problem. Our experimental analysis confirms a very good performance of the Metropolis algorithm, but suggests that the runtimes of the EDAs suffer from large constants hidden by the asymptotic analysis.

To avoid a possible misunderstanding, let us stress that our target is to find an established search heuristic for our optimization problem. From the above discourse one could believe that we should simply stick to the artificial black-box algorithm that we found. If our only aim was solving the DLB problem, this would indeed be feasible. Such an algorithm, however, would most likely lack the known positive properties of established search heuristics such as robustness to noise and dynamic changes of the problem data, reusability for similar problems, and adjustability to restricted instance classes. For that reason, our target in this work is definitely to find an established heuristic and not an particular algorithm custom-tailored to a problem.

We note that the results and methods used in this works lie purely in the theory domain. We therefore followed the traditional approach [DJW02] of regarding benchmark problems simple enough that they can be rigorously analyzed with mathematical means. In return, we obtain proven results for infinite numbers of problem instances (here, the DLB problem for all problem dimensions $n \in 2\mathbb{N}$), which hopefully extend in spirit also to problems which are too complicated to be analyzed with mathematical means.

We believe that our approach, in principle and in a less rigorous way, can also be followed by researchers and practitioners outside the theory community. Our basic approach of (i) trying to find a very good algorithm, chosen from all possible black-box optimization algorithms, to solve a given problem or to overcome a particular difficulty and then (ii) using this artificial and problem-specific algorithm as indicator for which established search heuristics could be promising, can also be followed by experimental methods and by non-rigorous intuitive considerations.

The remainder of the paper is organized as follows. Section 2 discusses black-box optimization, black-box complexity, the DLB problem, and two probabilistic tools to be used later. Sections 3 and 4 demonstrate our approach on the DLB problem. We first observe that non-elitist (1+1) type algorithms in principle can improve the $O(n^3)$ runtime of known elitist algorithms to a quadratic runtime and find the Metropolis algorithm as an

established heuristic showing this performance. By going beyond unary unbiased algorithms, we then obtain a further improvement to a complexity of $O(n \log n)$, first via an artificial algorithm and then via the significance-based compact genetic algorithm. Our experimental discussion is shown in Section 5, and Section 6 concludes this paper.

2 Preliminaries

Following the standard notation, we write $[\ell..m] := \{\ell, \ell+1, \dots, m\}$ for all $\ell, m \in \mathbb{N}$ such that $\ell \leq m$. In this paper we consider pseudo-Boolean optimization problems, that is, problems of *maximizing* functions f defined on the search space $\{0,1\}^n$, where n is a positive integer. A bit string (also called a search point) is an element of the set $\{0,1\}^n$ and is represented as $x = (x_1, \dots, x_n)$. In the context of evolutionary algorithms, we sometimes refer to a bit string as an individual and we use $x^{(t)}$ to denote an individual at time t, in which case $x_i^{(t)}$ is used to represent the i-th component of the individual $x^{(t)}$. To simplify the notation, whenever we write m where an integer is required, we implicitly mean $\lceil m \rceil := \min\{k \in \mathbb{N} \mid k \geq m\}$.

2.1 Black-Box Optimization and Runtime

In (discrete) black-box optimization, we assume that the optimization algorithms do not have access to an explicit description of the instance of the problem to be solved. Instead, their only access to the instance is via a black-box evaluation of search points. Classic black-box optimization algorithms include hill-climbers, the Metropolis algorithm, simulated annealing, evolutionary algorithms, and other bio-inspired search heuristics.

A general scheme for a black-box algorithm A is given in Algorithm 1. It starts by generating a random search point according to a given probability distribution and evaluating it. It then repeats generating (and evaluating) new search points based on all information collected so far, that is, based on all previous search points together with their fitnesses. While in practice, naturally, this iterative procedure is stopped at some time, in theoretical investigations it is usually assumed that this loop is continued forever. In this case, the *runtime*, also called *optimization time*, $T := T_A(f)$ of the algorithm A on the problem instance $f : \{0,1\}^n \to \mathbb{R}$ is the number of search points evaluated until (and including) for the first time an optimal solution of f is generated (and evaluated). In the notation of Algorithm 1, we have $T_A(f) = 1 + \inf\{t \mid x^{(t)} \in \arg\max f\}$. If the algorithm A is randomized (which most black-box optimizers are), then the runtime T is a random

variable and often only its expectation $E[T]$ is analyzed. For an *optimization problem*, that is, a set $\mathcal{F}$ of problem instances f, the *worst-case expected optimization time* $\sup_{f\in\mathcal{F}} E[T_A(f)]$ is an often regarded quality measure of the algorithm A.

Most black-box algorithms for practical reasons do not exploit this scheme to its full generality, that is, they do not store all information collected during the process or they do not sequentially generate one search point after the other based on all previous information, but instead generate in parallel several search points based on the same set of information. For our analyzes, we nevertheless assume that the search points are generated in some specified order and that *each search point is evaluated immediately after being generated*.

Algorithm 1: Template of a black-box algorithm for optimizing an unknown function $f : \{0,1\}^n \to \mathbb{R}$. Without explicit mention, we assume that each search point $x^{(t)}$ is evaluated immediately after being generated. As runtime of such an (infinitely running) algorithm we declare the number of search points evaluated until an optimum of f is evaluated for the first time.

```
1 Generate a search point x^(0) according to a given distribution D^(0)
    on {0,1}^n;
2 for t = 1, 2, 3, ... do
3 |  Depending on f(x^(0)), ..., f(x^(t−1)) and x^(0), ..., x^(t−1), choose a
  |   probability distribution D^(t) on {0,1}^n ;
4 |  Sample x^(t) from D^(t);
```

2.2 Unbiasedness

Unless a specific understanding of the problem at hand suggests a different approach, it is natural to look for optimization algorithms that are invariant under the symmetries of the search space. This appears so natural that, in fact, most algorithms have this property without that this has been discussed in detail.

The first to explicitly discuss such invariance properties for the search space $\{0,1\}^n$ of bit strings (see, e.g., [TGM06] for such a discussion for continuous search spaces) were Lehre and Witt in their seminal paper [LW12]. They coined the name *unbiased* for algorithms respecting the symmetries of the hypercube $\{0,1\}^n$. Such algorithms treat the bit positions $i \in [1..n]$ in a

symmetric fashion and, for each bit position, do not treat the value 0 differently from the value 1. It follows that all decisions of such algorithms may not depend on the particular bit string representation of the search points they have generated before. Rather, all decisions can only depend on the fitnesses of the search points generated. This implies that all search points the algorithm has access to can only be generated from previous ones via unbiased variation operators. This observation also allows to rigorously define the arity of an algorithm as the maximum number of parents used to generate offspring. Hence mutation-based algorithms have an arity of one and crossover-based algorithms have an arity of two. We note that sampling a random search point is an unbiased operator with arity zero.

Since these are important notions that also help to classify different kinds of black-box algorithms, let us make them precise in the following.

Definition 1. *A k-ary variation operator V is a function that assigns to each k-tuple of bit strings in $\{0,1\}^n$ a probability distribution on $\{0,1\}^n$. It is called* unbiased *if*

- $\forall x^1, \dots, x^k \in \{0,1\}^n, \forall y, z \in \{0,1\}^n$,
  $\Pr[y = V(x^1, \dots, x^k)] = \Pr[y \oplus z = V(x^1 \oplus z, \dots, x^k \oplus z)]$,
- $\forall x^1, \dots, x^k \in \{0,1\}^n, \forall y \in \{0,1\}^n, \forall \sigma \in \mathcal{S}_n$,
  $\Pr[y = V(x^1, \dots, x^k)] = \Pr[\sigma(y) = V(\sigma(x^1), \dots, \sigma(x^k))]$,

where $\mathcal{S}_n$ represents the symmetric group on n letters and we write $\sigma(x) = (x_{\sigma(1)}, \dots, x_{\sigma(n)})$ for all $\sigma \in \mathcal{S}_n$ and $x \in \{0,1\}^n$.

By definition, a k-ary operator can simulate ℓ-ary operators if $\ell \leq k$. It is also immediate that the only 0-ary operator is the operator that generates a search point in $\{0,1\}^n$ uniformly at random. As a special case, 1-ary unbiased variation operators are more often called unary unbiased variation operators and sometimes referred to as mutation operators in the context of evolutionary computation.

Unary unbiased variation operators admit a simple characterization, namely that sampling from the unary unbiased operator is equivalent to sampling a number $k \in [0..n]$ from some distribution and then flipping exactly k bits chosen uniformly at random. This is made precise in the following lemma, which was proven in [DDY20, Lemma 1], but which can, in a more general form, already be found in [DKLW13].

Lemma 2. *Let D be a probability distribution on $[0..n]$. Let V_D be the unary variation operator which for each $x \in \{0,1\}^n$ generates $V_D(x)$ by first sampling a number k from D and then flipping k random bits in x. Then V_D is a unary unbiased variation operator.*

Conversely, let V be a unary unbiased variation operator on $\{0,1\}^n$. Then there is a probability distribution D_V on $[0..n]$ such that $V = V_{D_V}$.

Building on the notion of a k-ary unbiased variation operator, we can now define what is a k-ary unbiased black-box algorithm (Algorithm 2). For the reasons given at the beginning of this section, in this work we shall only be interested in unbiased algorithms (possibly with unrestricted arity).

Algorithm 2: Template of a k-ary unbiased black-box algorithm for optimizing an unknown function $f : \{0,1\}^n \to \mathbb{R}$.

```
Generate x^(0) uniformly at random;
for t = 1, 2, 3, ... do
    Based solely on (f(x^(0)), ..., f(x^(t-1))), choose a k-ary unbiased
      variation operator V and i_1, ..., i_k ∈ [0..t-1];
    Sample x^(t) from V(x^(i_1), ..., x^(i_k));
```

2.3 Black-Box Complexity

To understand the difficulty of a problem for black-box optimization algorithms, inspired by classical complexity theory, Droste, Jansen, and Wegener [DJW06] (preliminary version [DJTW02], see also [Doe20d] for a recent survey) defined the *black-box complexity* of a problem $\mathcal{F}$ as the smallest worst-case expected runtime a black-box algorithm A can have on $\mathcal{F}$, that is,

$$\inf_A \sup_{f \in \mathcal{F}} E[T_A(f)],$$

where A runs over all black-box algorithms in the infimum.

Just by definition, the black-box complexity is a universal lower bound on the performance of any black-box algorithm. The result that black-box complexity of the Needle problem is $2^{n-1} + 0.5$ [DJW06, Theorem 1] immediately implies that no hillclimber, evolutionary algorithm, ant colony optimizer, etc. can solve the Needle problem in subexponential expected time.

Conversely, the black-box complexity can also serve as a trigger to search for better black-box algorithms. For example, in [DDE15] the observation that the black-box complexity of the OneMax problem is only $\Theta(n/\log n)$ [ER63, DJW06, AW09] (albeit witnessed by a highly artificial algorithm) whereas most classic evolutionary algorithms have a $\Theta(n \log n)$ runtime or

worse, was taken as starting point to design a novel evolutionary algorithm solving the OneMax problem in time asymptotically better than $n \log n$. This algorithm, called $(1 + (\lambda, \lambda))$ GA, has shown a good performance both in experiments [GP14, MB17, BD17] and in other mathematical analyses [BD17, DD18, ADK19, ADK20, ABD20a, ABD20b, AD20, ABD21].

In this work, as discussed in the introduction, we shall also use the black-box complexity as a trigger towards more efficient solutions to black-box optimization problems, however, not by suggesting new algorithms, but by suggesting the general type of algorithm that might be most suited for the problem and thus helping to choose the right algorithm among the many existing black-box algorithms. Compared to triggering the design of new algorithms, this might be the more effective road for people "merely" applying black-box algorithms. In fact, we shall argue that this road, despite the theory-based notion of black-box complexity, is in fact not too difficult to follow also for people without a background in algorithm theory.

2.4 The DLB Function and Known Runtimes

We now define the DLB function, which is the main object of our study and was first introduced by Lehre and Nguyen in their recent work [LN19].

To define the DLB function, the n-bit string x is divided, in a left-to-right fashion, into $\frac{n}{2}$ blocks of size of 2. The function value of x is determined by the longest prefix of 11 blocks and the following block. The blocks with two 1s in the prefix contribute each a value of 2 to the fitness. The DLB function is deceptive in that the next block contributes a value of 1 when it contains two 0s, but contributes only 0 when it contains one 1 and one 0. The optimum is the bit string with all 1s.

Since there is an apparent similarity between the DLB function and the classic LeadingOnes benchmark function, we first recall the definition of the latter. The LeadingOnes function (which shall be abbreviated as LO in what follows) is defined for all $x \in \{0,1\}^n$ by $\mathrm{LO}(x) = \sum_{r \in [1..n]} \prod_{s \in [1..r]} x_s$.

The DLB function is defined on $\{0,1\}^n$ only for n even. We therefore assume in the following that n is even whenever this is required.

For all $x \in \{0,1\}^n$, we formally define the DLB function in the following way. We consider blocks of the form $(x_{2\ell+1}, x_{2\ell+2})$. If $x \neq (1, \dots, 1)$, then let (x_{2m+1}, x_{2m+2}) be the first block that is not a 11 block, that is, $m = \inf\{\ell \mid x_{2\ell+1} \neq 1 \text{ or } x_{2\ell+2} \neq 1\}$. We call such a block a *critical block*. Then the DLB

function is defined via

$$\mathrm{DLB}(x) = \begin{cases} 2m+1 & \text{if } x_{2m+1} + x_{2m+2} = 0, \\ 2m & \text{if } x_{2m+1} + x_{2m+2} = 1, \\ n & \text{if } x = (1, \dots, 1). \end{cases}$$

In other words, the DLB function counts twice the number of consecutive 11 blocks until it reaches a critical block, which counts for 1 if it is of the form 00 and counts for 0 if it is of the form 01 or 10. Hence the search points $x = (1, \dots, 1, 0, 0, x_{2\ell+1}, \dots, x_n)$ with $\ell \in [1..\frac{n}{2}]$ and $x_{2\ell+1}, \dots, x_n \in \{0, 1\}$ are the local maxima of the DLB function. The unique global maximum of the DLB function is $x^* = (1, \dots, 1)$.

Right from the definitions, we see that the functions DLB and LO are very similar, indeed, we have $|\mathrm{LO}(x) - \mathrm{DLB}(x)| \leq 1$ for all $x \in \{0, 1\}^n$. The main difference is that DLB has non-trivial local optima which could render its optimization harder.

In a similar fashion we define the *Honest Leading Blocks* (HLB) function, which will be used as potential function in some proofs using drift analysis. The DLB function being deceptive and unable to discern a 01 critical block from a 10 critical block, the HLB function also treats 01 and 10 critical blocks equally, but is honest with the fact that 01 and 10 critical blocks are better than a 00 critical block in the sense that such search points are closer to the global maximum $(1, \dots, 1)$. Formally speaking, the HLB function with parameter $\delta \in (0, 2)$ is defined by

$$\mathrm{HLB}_\delta(x) := \begin{cases} 2m & \text{if } \mathrm{DLB}(x) = 2m+1, \\ 2m+2-\delta & \text{if } \mathrm{DLB}(x) = 2m, \\ n & \text{if } \mathrm{DLB}(x) = n, \end{cases}$$

where m is an integer in $\{0, 1, \dots, \frac{n}{2} - 1\}$.

We now review the most relevant known runtime results for this work. Lehre and Nguyen [LN19] proved that, always assuming that the mutation rate used by the algorithm is χ/n for a constant $\chi > 0$, the expected runtime of the $(1+\lambda)$ EA on the DLB problem is $O(n\lambda + n^3)$ and that the expected runtime of the $(\mu+1)$ EA on the DLB problem is $O(\mu n \log n + n^3)$. They also proved that the expected runtime of the (μ, λ) EA on the DLB function is $O(n\lambda \log \lambda + n^3)$ under the conditions that for an arbitrary constant $\delta > 0$ and a constant c sufficiently large, we have $\lambda > c \log n$ and $\mu < \frac{\lambda e^{-2\chi}}{1+\delta}$. Furthermore, they showed that, with a good choice of the parameters, genetic algorithms using k-tournament selection, (μ, λ) selection, linear selection, or exponential ranking selection, also have an $O(n\lambda \log \lambda + n^3)$ expected runtime on the DLB problem.

From looking at the proofs in [LN19], it appears natural that all algorithms given above have a runtime of at least $\Omega(n^3)$ on the DLB problem, but the only proven such result is that Doerr and Krejca [DK20b] showed that the $(1+1)$ EA with mutation rate $1/n$ solves the DLB problem in $\Theta(n^3)$ expected fitness evaluations. In Theorem 10, we extend this result to all $(1+1)$-elitist unary unbiased black-box algorithms.

As opposed to these polynomial runtime results, Lehre and Nguyen pointed out in [LN19] a potential weakness of the Univariate Marginal Distribution Algorithms (UMDA). They proved that the UMDA selecting μ fittest individuals from λ sampled individuals has an expected runtime of $e^{\Omega(\mu)}$ on the DLB problem if $\frac{\mu}{\lambda} \geq \frac{14}{1000}$ and $c \log n \leq \mu = o(n)$ for some sufficiently large constant $c > 0$, and has expected runtime $O(n\lambda \log \lambda + n^3)$ if $\lambda \geq (1+\delta)e\mu^2$ for any $\delta > 0$. However, Doerr and Krejca [DK20b] pointed out that this negative finding can be overcome with a different parameter choice and that with a population size large enough to prevent genetic drift [SW19, DZ20], the UMDA solves the DLB problem efficiently. To be precise, they proved that the UMDA optimizes the DLB problem within probability at least $1 - \frac{9}{n}$ within $\lambda\left(\frac{n}{2} + 2e \ln n\right)$ fitness evaluations if $\mu \geq c_\mu n \ln n$ and $\mu/\lambda \leq c_\lambda$ for some c_μ, c_λ sufficiently large or small, respectively.

Since the DLB function bears some similarity with the LeadingOnes problem, it will be useful to compare the runtimes on these two problems. The LeadingOnes problem was proposed in [Rud97] as an example of a unimodal problem which simple EAs cannot solve in $O(n \log n)$ time. The correct asymptotic runtime of the $(1+1)$ EA of order $\Theta(n^2)$ was determined in [DJW02]. The precise runtime of the $(1+1)$ EA was independently determined in [BDN10, Sud13]. Precise runtimes of various other variants of the $(1+1)$ EA were given in [Doe19a]. The runtime of the $(\mu+\lambda)$ EA with μ at most polynomial in n on LeadingOnes is $\Theta(\mu n \log n + n^2)$ [Wit06], the one of the $(1+\lambda)$ EA with λ at most polynomial in n is $\Theta(\lambda n + n^2)$. For the $(\mu+\lambda)$ EA, only a lower bound of $\Omega(\frac{\lambda n}{\log(\lambda/n)})$ is known [BLS14, LS20]. The runtime of the $(1+(\lambda,\lambda))$ GA with standard parameterization $p = \lambda/n$ and $c = 1/\lambda$ is $\Theta(n^2)$ regardless of the value of $\lambda \in [1..n/2]$ and this also with dynamic parameter choices [ADK19]. Consequently, for a large number of elitist algorithms, the runtime is $\Theta(n^2)$ when suitble parameters are used. In [DL18], Doerr and Lengler have shown that all $(1+1)$-elitist algorithms need $\Omega(n^2)$ fitness evaluations in expectation to solve the LeadingOnes problem. This result implies the lower bounds in [DJW02, BDN10, Sud13, Doe19a] when ignoring constant factors.

For non-elitist algorithms, the picture is less clear. Upper bounds have been shown for various algorithms, also some using crossover, when the se-

lection pressure is high enough [Leh11, DL16, CDEL18, DK19], but none of them beats the $\Omega(n^2)$ barrier. When the selection pressure is small, many non-elitist algorithm cannot optimize any function with unique optimum in subexponential time [Leh10, Doe21].

Upper bounds were also shown for the runtime of the estimation-of-distribution algorithms UMDA and PBIL in [DLN19, LN18] and for the ant-colony optimizers 1-ANT and MMAS in [DNSW11, NSW09], but again none could show a runtime better than quadratic in n.

A runtime better than quadratic, and in fact of order $O(n \log n)$ was shown for the three non-classical algorithms CSA [MS17], scGA [FKK16], and sig-cGA [DK20a]. The first two of these, however, are highly inefficient on the OneMax benchmark and thus might be overfitted to the LeadingOnes problem.

The unrestricted black-box complexity of the LeadingOnes class is $\Theta(n \log \log n)$, as witnessed by a highly problem-specific algorithm in [AAD+19].

2.5 Probabilistic Tools

We now collect two probabilistic tools used in the remainder of this work. The additive drift theorem is commonly used to derive upper (resp. lower) bounds on the expected runtime of an algorithm from lower (resp. upper) bounds on the expected increase of a suitable potential. It first appeared in the analysis of evolutionary algorithms in He and Yao's work [HY01, HY04], in which they implicitly used the optional stopping theorem for martingales. The following version can be found in Lengler's survey [Len20]. It was first proven in Lengler and Steger's work [LS18] via an approach different from the one He and Yao used.

Theorem 3 ([Len20], Theorem 2.3.1)**.** *Let $(h_t)_{t \geq 0}$ be a sequence of non-negative random variables taking values in a finite set $\mathcal{S} \subseteq [0, n]$ such that $n \in \mathcal{S}$. Let $T := \inf\{t \geq 0 \mid h_t = n\}$ be the first time when $(h_t)_{t \geq 0}$ takes the value n. For all $s \in \mathcal{S}$, let $\Delta_t(s) := E[h_{t+1} - h_t \mid h_t = s]$. Then the following two assertions hold.*

- *If for some $\delta > 0$ we have $\Delta_t(s) \geq \delta$ for all $s \in \mathcal{S} \setminus \{n\}$ and all t, then $E[T] \leq E[n - h_0]\delta^{-1}$.*
- *If for some $\delta > 0$ we have $\Delta_t(s) \leq \delta$ for all $s \in \mathcal{S} \setminus \{n\}$ and all t, then $E[T] \geq E[n - h_0]\delta^{-1}$.*

Chernoff-Hoeffding inequalities [Che52, Hoe63], often just called *Chernoff bounds*, are a standard tool in the analysis of random structures and

algorithms. In this work we will use the following variance-based additive Chernoff bound (see e.g. [Doe20c, Theorem 1.10.12]).

Theorem 4. *(variance-based additive Chernoff inequality) Let $X_1, \dots, X_n$ be independent random variables and suppose that for all $i \in [1..n]$, we have $|X_i - E[X_i]| \leq 1$. Let $X := \sum_{i=1}^{n} X_i$ and $\sigma^2 := \mathrm{Var}[X] = \sum_{i=1}^{n} \mathrm{Var}[X_i]$. Then for all $\lambda \in (0, \sigma^2)$,*

$$\Pr[X \geq E[X] + \lambda] \leq e^{-\frac{\lambda^2}{3\sigma^2}} \text{ and } \Pr[X \leq E[X] - \lambda] \leq e^{-\frac{\lambda^2}{3\sigma^2}}.$$

3 From Elitist to Non-Elitist Algorithms

Previous works have shown that the expected runtime of the (1+1) EA on the DLB problem is $\Theta(n^3)$, see [LN19, Theorem 3.1] for the upper and [DK20c, Theorem 4] for the lower bound (following from the precise computation of the expected runtime there).

In this section, we extend this lower bound and show that any $(1 + 1)$-elitist unary unbiased black-box algorithm has a runtime of at least $\Omega(n^3)$. This result will motivate us to study non-elitist (1+1)-type algorithms, which will lead to the discovery that the Metropolis algorithm can solve the DLB problem significantly faster.

3.1 Elitist Algorithms Suffer From the Fitness Valleys

3.1.1 $(\mu + \lambda)$-Elitist Black-Box Complexity

We start by making precise the elitist black-box complexity model we regard. Since it might be useful in the future, we first define our model for general $(\mu + \lambda)$-elitism, even though our main result in this section only considers $(1 + 1)$-elitist algorithms.

A $(\mu + \lambda)$-elitist algorithm uses a parent population of size μ. In each iteration, it generates from it λ offspring and determines the next parent population by choosing μ best individual from the $\mu + \lambda$ parents and offspring. Hence the term "elitist" refers to the restriction that the next parent population has to consist of μ best individuals. Ties can be broken arbitrarily, in particular, in case of ties there is no need to prefer offspring.

We shall further only regard algorithms that are unbiased in the sense of Lehre and Witt [LW12] (see Section 2.2). This in particular means that the algorithm has never access to the bit string representation of individuals (except from using unbiased variation operators and computing the fitness). Consequently, all choices done by the algorithm such as choosing parents for

the creation of offspring, choosing variation operators, and selecting the next parent population can only rely on the fitnesses of the individuals created so far. Finally, as variation operators we shall only allow unary (mutation) operators (see Definition 1).

In summary, we obtain the $(\mu + \lambda)$-elitist unary unbiased black-box algorithm class described in Algorithm 3. It is similar to the $(\mu + \lambda)$-elitist model proposed in [DL17], but differs from it in that we do not require that the algorithm only has access to a ranking of the search points and in that we do require the algorithm to be unary unbiased. Nevertheless, to ease the language, we shall in the remainder call our algorithms simply $(\mu + \lambda)$-elitist algorithms, that is, we suppress the explicit mention of the unary unbiasedness.

Algorithm 3: Template of a $(\mu + \lambda)$-elitist unary unbiased black-box algorithm, $(\mu + \lambda)$-elitist algorithm for short, for optimizing an unknown function $f : \{0, 1\}^n \to \mathbb{R}$.

```
Generate μ search points x^(0,i), i ∈ [1..μ], independently and
  uniformly at random;
X ← {x^(0,i) | i ∈ [1..μ]};
for t = 1, 2, 3, ... do
    Choose λ individuals p_1, ..., p_λ from X;
    Choose λ unary unbiased operators V_1, ..., V_λ;
    Sample q_1, ..., q_λ from V_1(p_1), ..., V_λ(p_λ) respectively;
    X ← a selection of μ best individuals from X ∪ {q_1, ..., q_λ};
```

For a $(\mu+\lambda)$-elitist algorithm A, we recall that the runtime $T_A(f)$ on the maximization problem f is by definition the number of fitness evaluations performed until a maximum of f is evaluated for the first time. The $(\mu+\lambda)$-elitist black-box complexity of the optimization problem of f is defined to be

$$\inf_A E[T_A(f)],$$

where A runs through all $(\mu + \lambda)$ unary unbiased black-box algorithms (Algorithm 3).

3.1.2 Independence of Irrelevant Bits

We start our analysis with a simple lemma showing that bit positions that did not have an influence on any fitness evaluation are independently and uniformly distributed. This lemma is similar to [LW12, Lemma 1].

Lemma 5. *Let $f : \{0,1\}^n \to \mathbb{R}$, $c \in \operatorname{Im} f$, and $I \subset [1..n]$. Let $Z = \{z \in \{0,1\}^n \mid \forall i \notin I : z_i = 0\}$. Assume that for all $y \in \{0,1\}^n$ with $f(y) \leq c$, we have $f(y \oplus z) = f(y)$ for all $z \in Z$.*

Then for any $(\mu+\lambda)$-elitist algorithm and any $k \in [1..\mu]$, conditioning on the event

$$E_{t,c} := \{\max_{j \in [1..\mu]} f(x^{(t,j)}) = c\},$$

the bits $x_i^{(t,k)}$, $i \in I$, are all independent and uniformly distributed in $\{0,1\}$.[1]

Proof. We consider the joint distribution of the random variables $(x^{(s,j)})_{s \in [0..t], j \in [1..\mu]}$ under the condition $E_{t,c}$. By the hypothesis and the mechanisms of $(\mu + \lambda)$-elitist algorithms, $(x^{(s,j)})_{s \in [0..t], j \in [1..\mu]}$ and $(x^{(s,j)} \oplus z)_{s \in [0..t], j \in [1..\mu]}$ are identically distributed for any $z \in Z$.

In particular, for any $k \in [1..\mu]$ and any $z \in Z$, $x^{(t,k)}$ and $x^{(t,k)} \oplus z$ are identically distributed. From this we deduce that for any $x \in \{0,1\}^n$,

$$\Pr[x^{(t,k)} = x] = 2^{-|I|} \Pr[x^{(t,k)}_{|[1..n] \setminus I} = x_{|[1..n] \setminus I}].$$

Therefore, under the condition $\{x_i^{(t,k)} = x_i, \forall i \in [1..n] \setminus I\}$ where x_i, $i \in [1..n] \setminus I$, are prescribed bit values, the bits $x_i^{(t,k)}$, $i \in I$, are all independent and uniformly distributed in $\{0,1\}$, which implies the claim. □

We apply the above lemma to the optimization of the DLB function.

Lemma 6. *Let $m \in [0..\frac{n}{2} - 1]$. For any $(\mu + \lambda)$-elitist algorithm and any $k \in [1..\mu]$, conditioning on the event*

$$\left\{\max_{j \in [1..\mu]} \mathrm{DLB}(x^{(t,j)}) = 2m\right\},$$

or on the event

$$\left\{\max_{j \in [1..\mu]} \mathrm{DLB}(x^{(t,j)}) = 2m+1\right\},$$

the bits $x_i^{(t,k)}$, $i = [2m+3..n]$, are all independent and uniformly distributed in $\{0,1\}$.

Proof. It suffices to take $c = 2m$ ($c = 2m + 1$ for the second case) and $I = [2m + 3..n]$ in the preceding lemma. □

[1] We use this language here and in the remainder to express that the bits x_i, $i \in i$, are mutually independent, independent of all other bits of x, and uniformly distributed in $\{0,1\}$.

3.1.3 Runtime Analysis of $(1+1)$ Elitist Unary Unbiased EAs

This section is devoted to proving that the (1+1)-elitist black-box complexity of the DLB problem is $\Omega(n^3)$. To this end we use drift analysis and take the HLB function as potential.

We start with an estimate of the influence of the so-called free-riders, that is, we estimate that the expected potential of a random string with $2m+2$ leading ones is at most $2m+4$.

Lemma 7. *Let $m \in [0..\frac{n}{2}-1]$. Let x be a random bit string such that $x_i = 1$ for all $i \in [1..2m+2]$ and such that the bits x_i, $i \in [2m+3..n]$ are independent and uniformly distributed in $\{0,1\}$. Then for any $\delta \in [0,2]$ we have*

$$E[\mathrm{HLB}_\delta(x)] \leq 2m+4.$$

Proof. The statement clearly holds for $m = \frac{n}{2}-1$ since in this case $x = (1,\dots,1)$ and $\mathrm{HLB}_\delta((1,\dots,1)) = n < n+2$. Now we proceed by backwards induction on m. Suppose that the conclusion holds for all $m = k+1,\dots,\frac{n}{2}-1$. For $m = k$ we compute

$$\begin{aligned} &E[\mathrm{HLB}_\delta(x)] \\ &\quad = \tfrac{1}{4}(2k+2) + \tfrac{1}{2}(2k+4-\delta) + \tfrac{1}{4}E[\mathrm{HLB}_\delta(x) \mid x_{2k+3} = x_{2k+4} = 1]. \end{aligned}$$

The induction hypothesis can be applied to the last term, yielding

$$E[\mathrm{HLB}_\delta(x)] \leq \tfrac{1}{4}(2k+2) + \tfrac{1}{2}(2k+4-\delta) + \tfrac{1}{4}(2k+6) \leq 2k+4.$$

By induction, this proves the lemma. □

We now estimate the expected progress in one iteration, first in the case that the parent has an even DLB value (Lemma 8), then in the case that it is a local optimum (Lemma 9).

Lemma 8. *Let $m \in [\frac{n}{4}..\frac{n}{2}-1]$. Let x be a random bit string such that $\mathrm{DLB}(x) = 2m$ and that the bits x_i, $i \in [2m+3..n]$ are independent and uniformly distributed in $\{0,1\}$. Let y be a random bit string generated from x via a unary unbiased variation operator V. Let $Y := (\mathrm{HLB}_\delta(y) - \mathrm{HLB}_\delta(x))\mathbb{1}_{\mathrm{DLB}(y)\geq\mathrm{DLB}(x)}$. Then we have*

$$E[Y] \leq \frac{2\delta}{n}.$$

Proof. Using Lemma 2, we can decompose the expectation by conditioning on the value of r_V:

$$E[Y] = \sum_{r=1}^{n} E[Y \mid r_V = r] \Pr[r_V = r]. \tag{1}$$

By symmetry we assume that $x_{2m+1} = 1$ and $x_{2m+2} = 0$. We use F_{2m+1} (F_{2m+2} resp.) to denote the event in which x_{2m+1} (x_{2m+2} resp.) is the only bit that has been flipped among the first $2m+2$ bits of x. We observe that $\{Y < 0\} = F_{2m+1}$ and $\{Y > 0\} = F_{2m+2}$. It follows that

$$\begin{aligned}
&E[Y \mid r_V = r] \\
&\quad = E[Y \mid r_V = r, F_{2m+1}] \Pr[F_{2m+1} \mid r_V = r] \\
&\qquad + E[Y \mid r_V = r, F_{2m+2}] \Pr[F_{2m+2} \mid r_V = r] \\
&\quad = (\delta - 2) \Pr[F_{2m+1} \mid r_V = r] \\
&\qquad + E[\mathrm{HLB}_\delta(y) - (2m + 2 - \delta) \mid r_V = r, F_{2m+2}] \Pr[F_{2m+2} \mid r_V = r].
\end{aligned}$$

Conditioning on the event $\{r_V = r\} \cap F_{2m+2}$, the random bit string y has only 1s in its first $2m+2$ bit positions and the other bits of y are independent and uniformly distributed in $\{0, 1\}$. Lemma 7 thus gives that

$$E[\mathrm{HLB}_\delta(y) \mid r_V = r, F_{2m+2}] \leq 2m + 4.$$

Therefore we have

$$\begin{aligned}
&E[Y \mid r_V = r] \\
&\qquad \leq (\delta - 2) \Pr[F_{2m+1} \mid r_V = r] + (\delta + 2) \Pr[F_{2m+2} \mid r_V = r]. \qquad (2)
\end{aligned}$$

Now we analyze the two probabilities

$$\begin{aligned}
\Pr[F_{2m+1} \mid r_V = r] &= \Pr[F_{2m+2} \mid r_V = r] \\
&= \binom{n - 2m - 2}{r - 1} \binom{n}{r}^{-1} =: D_r. \qquad (3)
\end{aligned}$$

We calculate for $r - 1 \leq n - 2m - 2$ that

$$\begin{aligned}
\frac{D_{r+1}}{D_r} &= \frac{\binom{n-2m-2}{r} \binom{n}{r+1}^{-1}}{\binom{n-2m-2}{r-1} \binom{n}{r}^{-1}} = \frac{(r+1)(n-2m-1-r)}{r(n-r)} \\
&= \frac{n - 2m - 1 - 2(m+1)r}{r(n-r)} + 1.
\end{aligned}$$

Since $r \geq 1$ and $m \geq \frac{n}{4}$, we have

$$n - 2m - 1 - 2(m+1)r \leq n - 4m - 3 < 0.$$

This implies that $\frac{D_{r+1}}{D_r} < 1$. Noting also that $D_r = 0$ for $r > n - 2m - 2$, we see that D_r is decreasing in r. Consequently, we have

$$D_r \leq D_1 = \frac{1}{n} \qquad (4)$$

for all $r \geq 1$. Combining this with (2) and (3), we obtain

$$E[Y \mid r_V = r] \leq 2\delta D_r \leq \frac{2\delta}{n}$$

for all $r \geq 1$. Substituting this into (1), the conclusion follows. □

Lemma 9. *Let $m \in [\frac{n}{3}, \frac{n}{2} - 1]$. Let x be a random bit string such that $\mathrm{DLB}(x) = 2m + 1$ and that the bits x_i, $i \in [2m + 3..n]$, are independent and uniformly distributed in $\{0, 1\}$. Let y be a random bit string generated from x by a unary unbiased variation operator V. Let $Y := (\mathrm{HLB}_\delta(y) - \mathrm{HLB}_\delta(x))\mathbb{1}_{\mathrm{DLB}(y) \geq \mathrm{DLB}(x)}$. Then we have*

$$E[Y] \leq \frac{16}{n^2}.$$

Proof. Since $\mathrm{DLB}(x) = 2m + 1$, $\mathrm{DLB}(y) \geq \mathrm{DLB}(x)$ implies $\mathrm{HLB}_\delta(y) \geq \mathrm{HLB}_\delta(x)$. Thus Y is always non-negative and we can decompose $E[Y]$ into a product of two terms

$$E[Y] = E[\mathrm{HLB}_\delta(y) - \mathrm{HLB}_\delta(x) \mid Y > 0] \Pr[Y > 0]. \tag{5}$$

First we bound the term $E[\mathrm{HLB}_\delta(y) - \mathrm{HLB}_\delta(x) \mid Y > 0]$. Under the condition $Y > 0$, we know that $y_i = 1$ for $i \in [1..2m + 2]$, while the y_i, $i \in [2m + 3..n]$, are still independent and uniformly distributed in $\{0, 1\}$. Lemma 7 then implies that $E[\mathrm{HLB}_\delta(y) \mid Y > 0] \leq 2m + 4$, thus we have

$$E[\mathrm{HLB}_\delta(y) - \mathrm{HLB}_\delta(x) \mid Y > 0] \leq 2m + 4 - 2m = 4. \tag{6}$$

On the other hand, to bound $\Pr[Y > 0] = \Pr[\mathrm{DLB}(y) > \mathrm{DLB}(x)]$ we invoke Lemma 2 and compute

$$\begin{aligned}
&\Pr[\mathrm{DLB}(y) > \mathrm{DLB}(x)] \\
&= \sum_{r=2}^{n-2m} \Pr[r_V = r] \Pr[\mathrm{DLB}(y) > \mathrm{DLB}(x) \mid r_V = r] \\
&= \sum_{r=2}^{n-2m} \Pr[r_V = r] \binom{n-2m-2}{r-2} \binom{n}{r}^{-1} \\
&\leq \max_{2 \leq r \leq n-2m} \binom{n-2m-2}{r-2} \binom{n}{r}^{-1},
\end{aligned}$$

where r can be restricted to $2 \leq r \leq n-2m$ since otherwise Y would certainly be zero. With $E_r := \binom{n-2m-2}{r-2}\binom{n}{r}^{-1}$, we have

$$\frac{E_{r+1}}{E_r} = \frac{2(n - m - (m+1)r)}{(r-1)(n-r)} + 1 < 1$$

by our hypothesis $n - 3m \leq 0$ and $r \geq 2$. As in Lemma 8 we conclude that E_r is decreasing in r and

$$\begin{aligned}&\Pr[Y > 0]\\ &= \Pr[\mathrm{DLB}(y) > \mathrm{DLB}(x)] \leq \max_{2 \leq r \leq n-2m} E_r = E_2 = \binom{n}{2}^{-1} \leq \frac{4}{n^2}. \quad (7)\end{aligned}$$

Combining (5), (6) and (7), we obtain $E[Y] \leq \frac{16}{n^2}$. □

With the estimates on the change of the potential HLB above, we can now easily prove the main result of this section. By taking $\delta = \Theta(\frac{1}{n})$ suitably, we ensure that the expected gain of the potential in one iteration is only $O(\frac{1}{n^2})$, at least in a sufficiently large range of the potential space. To have a small progress in the whole search space, as necessary to apply the additive drift theorem, we regard the potential $h_t = \mathrm{HLB}_\delta(x^{(t+S)})$, where S denotes the first time that $x^{(t)}$ has an HLB value not less than $\frac{2n}{3}$. This potential also has a drift of only $O(\frac{1}{n^2})$ and thus the additive drift theorem easily gives the lower bound of $\Omega(n^3)$.

Theorem 10. *The* $(1+1)$*-elitist black-box complexity of the* DLB *problem is* $\Omega(n^3)$.

Proof. Consider any (1+1)-elitist algorithm A. Let $x^{(t)}$ denote the individual at time t. By a slight abuse of notation we write $\frac{2n}{3}$ to denote the even integer $\min\{\ell \in 2\mathbb{N} \mid \ell \geq \frac{2n}{3}\}$. First we observe that $\Pr\big[\mathrm{HLB}(x^{(0)}) \geq \frac{2n}{3}\big] = \Theta(2^{-\frac{2}{3}n})$. In the following we work under the condition $\mathcal{C} := \{\mathrm{HLB}(x^{(0)}) < \frac{2n}{3}\}$. Under this condition, $S := \min\{t \in \mathbb{N} \mid \mathrm{HLB}(x^{(t)}) \geq \frac{2n}{3}\}$ is strictly positive. Let $T := \min\{t \in \mathbb{N} \mid \mathrm{HLB}(x^{(t)}) \geq n\} + 1$ be the runtime of A. We will use the additive drift theorem (Theorem 3) to prove $E[T - S \mid \mathcal{C}] = \Omega(n^3)$, from which the claim follows via

$$E[T] \geq \Pr[\mathcal{C}] E[T \mid \mathcal{C}] \geq (1 - o(1)) E[T - S \mid \mathcal{C}]. \quad (8)$$

For a δ to be specified later, we regard the potential $h_t = \mathrm{HLB}_\delta(x^{(t+S)})$. Note that because of the elitism of our algorithm, we have $h_t \geq \frac{2n}{3}$ for all $t \geq 0$. Let $m \in [\frac{n}{3}..\frac{n}{2} - 1]$.

Lemma 6 implies that when conditioning on the event $\{h_t = 2m+2-\delta\} = \{\mathrm{DLB}(x^{(t+S)}) = 2m\}$, the bits $x_i^{(t+S)}$, $i \in [2m+3..n]$, are independent and uniformly distributed in $\{0,1\}$. Let y be a bit string obtained by applying a unary unbiased variation operator to $x^{(t+S)}$. Lemma 8 then applies to y and $x^{(t+S)}$, yielding

$$E\big[(\mathrm{HLB}_\delta(y) - \mathrm{HLB}_\delta(x^{(t+S)}))\mathbb{1}_{\mathrm{DLB}(y) \geq \mathrm{DLB}(x^{(t+S)})} \,\big|\, \mathcal{C}\big] \leq \frac{2\delta}{n}.$$

Since the term on the left-hand side is the expected increase in the HLB function during an iteration, we have in fact proven that $\Delta_t(2m+2-\delta) := E[h_{t+1} - h_t \mid h_t = 2m+2-\delta, \mathcal{C}] \leq \frac{2\delta}{n}$ for any t.

For the case $\{h_t = 2m\}$ we proceed in the same manner (except that we use Lemma 9 instead of Lemma 8) to obtain $\Delta_t(2m) \leq \frac{16}{n^2}$ for any t. By setting δ to $\frac{8}{n}$, $\Delta_t(s) \leq \frac{16}{n^2}$ holds for any possible h_t value s.

Recall that $\frac{2n}{3}$ denotes the smallest even integer not less than it. By the definition of S and by Lemma 6, $x^{(S)}$ is a random bit string satisfying $x_i^{(S)} = 1$ for $i \in [1..\frac{2n}{3}]$ and such that $x_i^{(S)}$, $i \in [\frac{2n}{3}+1..n]$, are independent and uniformly distributed in $\{0,1\}$. Thus we have $E[h_0] = E[\mathrm{HLB}_\delta(x^{(S)})] = \frac{2n}{3} + O(1)$ by virtue of Lemma 7. The additive drift theorem (Theorem 3) then yields

$$E[T - S \mid \mathcal{C}] \geq \frac{n - \frac{2n}{3} - O(1)}{16/n^2} = \Theta(n^3),$$

which finishes the proof with (8). □

3.2 The Unary Unbiased Black-box Complexity of the DLB Problem is at Most Quadratic

In the preceding section we have seen that elitism does not ease optimizing the DLB problem. This section, therefore, is devoted to investigating the best possible expected runtime on the DLB problem without elitism. To be more precise, we show that the unary unbiased black-box complexity of the DLB problem is $O(n^2)$. This result will be complemented by a matching lower bound (Theorem 17) in Section 3.5.

We recall the definition of a k-ary unbiased black-box algorithm (Algorithm 2) from Section 2.2. The k-ary unbiased black-box complexity of a problem is defined as the infimum expected runtime of a k-ary unbiased black-box algorithm on this problem, that is,

$$\inf_A E[T_A],$$

where A runs through all k-ary unbiased black-box algorithms.

Let V be the unary unbiased operator such that $V(x)$ is obtained from flipping one bit of x. Then the additive drift theorem implies that the (1+1)-elitist black-box algorithm using V as variation operator has an expected runtime of $O(n^2)$ on the LO problem. By a small adjustment of this $(1+1)$-elitist black-box algorithm, we exhibit a simple unary unbiased black-box algorithm that solves the DLB problem in expected time $\Theta(n^2)$. This inspires our investigation on the expected runtime of the Metropolis algorithm in Section 3.3.

Lemma 11. *The unary unbiased black-box complexity of the* DLB *problem is* $O(n^2)$.

Proof. We present a simple algorithm and then show that its expected runtime is indeed $O(n^2)$. Throughout the algorithm we only use the unary unbiased operator V that flips exactly one randomly chosen bit.

The algorithm is initialized by generating a search point $x^{(0)}$ at random in $\{0,1\}^n$. In generation t, we generate a bit string $y = V(x^{(t)})$. If $\mathrm{DLB}(x^{(t)})$ is even and $\mathrm{DLB}(x^{(t)}) < \mathrm{DLB}(y)$ we accept y as new search point, i.e., $x^{(t+1)} := y$. If $\mathrm{DLB}(x^{(t)})$ is odd and $\mathrm{DLB}(x^{(t)}) < 2 + \mathrm{DLB}(y)$, we also accept y as new search point, i.e., $x^{(t+1)} := y$. In all other cases, we reject y, i.e., $x^{(t+1)} := x^{(t)}$.

Now we show that this algorithm finds the optimum of the DLB problem in time $O(n^2)$. We define the potential at time t as $k_t = \mathrm{HLB}_{\frac{3}{2}}(x^{(t)})$. In the case where $k_t = 2m$, that is, $\mathrm{DLB}(x^{(t)})$ is odd, we have a 00 critical block and the probability that we flip one of its two zeros is $\frac{2}{n}$, yielding an expected gain of at least $\frac{2}{n} \cdot \frac{1}{2} = \frac{1}{n}$ in the potential. In the case where $k_t = 2m + \frac{1}{2}$, that is, $\mathrm{DLB}(x^{(t)})$ is even, the critical block contains exactly one zero and one one. If the bit with value 0 is flipped, we increase the potential from $2m + \frac{1}{2}$ to at least $2(m+1)$. If the bit with value 1 is flipped, we reduce the potential from $2m + \frac{1}{2}$ to $2m$. All other bit-flips are not accepted or do not change the potential. Consequently, the expected gain in potential is at least $\frac{1}{n}(2(m+1) - (2m + \frac{1}{2})) + \frac{1}{n}(2m - (2m + \frac{1}{2})) = \frac{1}{n}$. Since the potential needs to be increased by at most n, the additive drift theorem (Theorem 3) establishes $\frac{n}{1/n} = n^2$ as an upper bound for the expected runtime of this algorithm on the DLB function. □

We remark that the artificial algorithm defined in the proof is a $(1+1)$-type unbiased algorithm, therefore we have the following stronger result.

Theorem 12. *The* $(1+1)$*-type unbiased black-box complexity of the* DLB *problem is* $O(n^2)$.

3.3 The Metropolis Algorithm Performs Well on the DLB Problem

Inspired by the analysis in the preceding sections, we expect that certain non-elitist $(1+1)$-type unbiased search heuristics outperform elitist EAs on the DLB problem. In fact, we will prove now that the Metropolis algorithm (simulated annealing with a fixed temperature) can solve the DLB problem within $\Theta(n^2)$ fitness evaluations in expectation. This performance coincides

with the unary unbiased black-box complexity of the DLB function (Theorem 17).

The Metropolis algorithm is a simple single-trajectory optimization heuristic. In contrast to elitist algorithms like randomized local search or the $(1+1)$ EA, it can accept inferior solutions, however only with a small probability that depends on the degree of inferiority and an algorithm parameter $\alpha \in (1, \infty)$.

More precisely, the maximization version of the Metropolis algorithm works as follows. It starts with a random initial solution $x^{(0)}$. In each iteration $t = 1, 2, \dots$, it generates a random neighbor y of the current search point $x^{(t-1)}$. When working with a bit string representation (as in this work), such a neighbor is obtained from flipping in $x^{(t-1)}$ a single bit chosen uniformly at random. If $f(y) \geq f(x^{(t-1)})$, then the algorithm surely accepts y as new search point $x^{(t)} := y$. If y is inferior to $x^{(t-1)}$, it accepts y ($x^{(t)} := y$) only with probability $\alpha^{f(y)-f(x^{(t-1)})}$ and otherwise rejects it ($x^{(t)} := x^{(t-1)}$). We note that the probability $\alpha^{f(y)-f(x^{(t-1)})}$ for accepting an inferior solution is often written as $\exp((f(y)-f(x^{(t-1)}))/kT)$ for a "temperature parameter" kT, but clearly the two formulations are equivalent. The pseudocode for this algorithm is given in Algorithm 4.

Algorithm 4: Metropolis algorithm for maximizing a function $f : \{0,1\}^n \to \mathbb{R}$

```
Generate a search point x^(0) uniformly in {0,1}^n;
for t = 1, 2, 3, ... do
    Choose i ∈ [1..n] uniformly at random and obtain y from flipping
     the i-th bit in x^(t−1);
    if f(y) ≥ f(x^(t−1)) then
        x^(t) ← y;
    else
        Choose b ∈ {0,1} randomly with Pr[b = 1] = α^(f(y)−f(x^(t−1)));
        if b = 1 then
            x^(t) ← y;
        else
            x^(t) ← x^(t−1);
```

Now we show that the Metropolis algorithm with sufficiently large parameter α can solve the DLB problem in time quadratic in n. To this end, we show the following elementary lemma.

Lemma 13. *For all $\alpha > \sqrt{2}+1$,*

$$C(\alpha) := \frac{2}{\alpha}\left(\frac{1}{2} - 2\sum_{k=1}^{\infty} k\alpha^{-2k}\right)$$

is strictly positive.

Proof. We observe that

$$\alpha^{-2}\sum_{k=1}^{\infty} k\alpha^{-2k} = \sum_{k=1}^{\infty} k\alpha^{-2(k+1)} = \sum_{k=2}^{\infty}(k-1)\alpha^{-2k}.$$

Subtracting $\sum_{k=1}^{\infty} k\alpha^{-2k}$ from both sides of the equation yields

$$(\alpha^{-2}-1)\sum_{k=1}^{\infty} k\alpha^{-2k} = \sum_{k=2}^{\infty} -\alpha^{-2k} - \alpha^{-2} = -\sum_{k=1}^{\infty}\alpha^{-2k} = -\frac{\alpha^{-2}}{1-\alpha^{-2}},$$

which implies

$$\sum_{k=1}^{\infty} k\alpha^{-2k} = \frac{\alpha^{-2}}{(1-\alpha^{-2})^2} = \frac{\alpha^2}{(\alpha^2-1)^2}.$$

Hence we have

$$C(\alpha) = \frac{2}{\alpha}\left(\frac{1}{2} - 2\frac{\alpha^2}{(\alpha^2-1)^2}\right) = \frac{\alpha^4-6\alpha^2+1}{\alpha(\alpha^2-1)^2}.$$

For all $\alpha > \sqrt{2}+1$, we have $\alpha^4-6\alpha^2+1 > 0$, which implies $C(\alpha) > 0$. □

We now show the main result of this section that the Metropolis algorithm can optimize the DLB function in quadratic time if the selection pressure is sufficiently high, that is, α is a large enough constant.

Theorem 14. *The expected runtime of the Metropolis algorithm on the DLB problem is at most $\frac{n^2}{C(\alpha)}$, provided that the parameter α satisfies $\alpha > \sqrt{2}+1$.*

To prove this result, we need to argue that the negative effect of accepting solutions further away from the optimum is outweighed by the positive effect that a critical 00-block can be changed into a critical block 01 or 10 despite the fact that this decreases the DLB value. To achieve this, we design a suitable potential function, namely the HLB function introduced in Section 2.4 with parameter $\delta = \frac{3}{2}$ and show that each iteration (starting with a non-optimal search point) in expectation increases this potential by $\Omega(\frac{1}{n})$. With this insight, the additive drift theorem immediately gives the claim.

Proof of Theorem 14. We denote by $x^{(t)}$ the search point obtained at the end of iteration t. To apply drift analysis we take $\mathrm{HLB}_{\frac{3}{2}}$ as potential and abbreviate $\mathrm{HLB}_{\frac{3}{2}}(x^{(t)})$ by h_t. Recalling that the Metropolis algorithm generates each Hamming neighbor y of $x^{(t-1)}$ with equal probability $\frac{1}{n}$, but accepts this only with probability $\alpha^{\min\{0,f(y)-f(x^{(t-1)})\}}$, we compute for each $m \in [0..\frac{n}{2}-1]$ that

$$\begin{aligned}
&E\Big[h_t - h_{t-1} \,\Big|\, h_{t-1} = 2m + \frac{1}{2}\Big] \\
&\geq \frac{1}{n}\left(2m+2-\left(2m+\frac{1}{2}\right)\right) + \frac{1}{n}\left(2m-\left(2m+\frac{1}{2}\right)\right) \\
&\quad + \sum_{k=0}^{m-1} \frac{2}{n}\alpha^{2k-2m}\left(2k+\frac{1}{2}-\left(2m+\frac{1}{2}\right)\right) \\
&= \frac{1}{n} + \frac{2}{n}\sum_{k=1}^{m}\alpha^{-2k}(-2k) = \frac{2}{n}\left(\frac{1}{2} - 2\sum_{k=1}^{m} k\alpha^{-2k}\right)
\end{aligned}$$

and

$$\begin{aligned}
&E[h_t - h_{t-1} \mid h_{t-1} = 2m] \\
&\geq \frac{2}{n}\frac{1}{\alpha}\left(2m+\frac{1}{2}-2m\right) + \sum_{k=0}^{m-1}\frac{2}{n}\alpha^{2k-2m-1}\left(2k+\frac{1}{2}-2m\right) \\
&= \frac{2}{n\alpha}\frac{1}{2} + \sum_{k=1}^{m}\frac{2}{n\alpha}\alpha^{-2k}\left(\frac{1}{2}-2k\right) > \frac{2}{n\alpha}\left(\frac{1}{2} - 2\sum_{k=1}^{m} k\alpha^{-2k}\right).
\end{aligned}$$

We have shown in Lemma 13 that $\frac{1}{2} - 2\sum_{k=1}^{\infty} k\alpha^{-2k} > 0$ for all $\alpha > \sqrt{2}+1$. Thus for any $s \neq n$,

$$\begin{aligned}
&E[h_t - h_{t-1} \mid h_{t-1} = s] \\
&\geq \frac{2}{n\alpha}\left(\frac{1}{2} - 2\sum_{k=1}^{m} k\alpha^{-2k}\right) \geq \frac{2}{n\alpha}\left(\frac{1}{2} - 2\sum_{k=1}^{\infty} k\alpha^{-2k}\right) = \frac{C(\alpha)}{n}.
\end{aligned}$$

The additive drift theorem (Theorem 3) now implies that the expected runtime of Metropolis algorithm on the DLB problem is bounded by

$$\frac{n}{C(\alpha)/n} = \frac{n^2}{C(\alpha)},$$

which concludes the proof. □

3.4 Literature Review on the Metropolis Algorithm and Non-Elitist Evolutionary Algorithms

To put our results on the Metropolis algorithm into context, we now briefly survey the known runtime results on this algorithm and non-elitist evolutionary algorithms, which are surprisingly few compared to the state of the art for elitist search heuristics.

3.4.1 Metropolis Algorithm

We first note that the Metropolis algorithm is a special case of simulated annealing, which can be described as the Metropolis algorithm using a temperature that is decreasing over time, that is, the parameter α increases over time in the language of Algorithm 4. We refer to [DT12] for a survey of the main applied results and restrict ourselves to the known theory results. Here the survey of Jansen [Jan11], even though from 2011, still is a good source of information.

Already in 1988, Sasaki and Hajek [SH88] showed that the Metropolis algorithm (and also its generalization simulated annealing) has an at least exponential worst-case runtime on the maximum matching problem. On the positive side, the Metropolis algorithm with constant temperature (constant α) can compute good approximate solutions. Analogous results, however, have also been shown for the $(1+1)$ EA by Giel and Wegener [GW03]. Jerrum and Sorkin [JS98] showed that the Metropolis algorithm can solve the minimum bisection problem in quadratic time in random instances in the planted bisection model. Wegener [Weg05] provided a simple instance of the minimum spanning tree problem, which can be solved very efficiently by simulated annealing with a natural cooling schedule, but for which the Metropolis algorithm with any temperature needs an exponential time to find the optimum. A similar result for the traveling salesman problem was given by Meer [Mee07].

Jansen and Wegener [JW07] proved that the Metropolis algorithm with $\alpha \geq \varepsilon n$, that is, with a very small temperature in the classic language, optimizes the OneMax benchmark in time $O(n \log n)$, a runtime also known for many simple evolutionary algorithms [Müh92, JJW05, Wit06, AD21]. They further show that this runtime is polynomial if and only if $\alpha = \Omega(n/\log n)$. Consequently, in the classic language, only for very small temperatures (that is, with very low probabilities of accepting an inferior solution) the Metropolis algorithm is efficient on the OneMax benchmark. An extension of this characterization to arbitrary symmetric functions (that is, functions f that can be written as $f(x) = g(\text{OneMax}(x))$ for some $g : \mathbb{R} \to \mathbb{R}$) was given

in [KWW09]. While it is interesting that such a characterization could be obtained, the characerization itself remains hard to interpret. Jansen and Wegener [JW07] further designed several example problems showing different or similar runtimes of the Metropolis algorithm and the $(1+1)$ EA, among them one that can be solved by the Metropolis algorithm with $\alpha = n$, that is, again a very small temperature, in polynomial expected time, whereas the $(1+1)$ EA needs time $n^{\Omega(\log \log n)}$, and one for which the Metropolis algorithm with any parameter setting has an expected runtime of $\Omega(2^{0.275n})$, whereas the $(1+1)$ EA has an expected runtime of only $\Theta(n^2)$.

In their work on the Metropolis algorithm and the Strong Selection Weak Mutation (SSWM) algorithm, Oliveto, Paixão, Heredia, Sudholt, and Trubenová [OPH+18] proposed the Valley problem, which contains a fitness valley with descending slope of length ℓ_1 and depth d_1 and ascending slope of length ℓ_2 and height d_2. This valley is constructed onto a long path function, making this essentially a one-dimensional optimization problem (despite being defined on $\{0,1\}^n$). They proved rigorously that the Metropolis algorithm takes an expected number of $n\alpha^{\Theta(d_1)} + \Theta(n\ell_2)$ function evaluations to cross this valley of low fitness when α (in the notation of Algorithm 4, note that the α used in [OPH+18] has a different meaning) is at least $\alpha \geq \exp(c \max\{\ell_1/d_1, \ell_2/d_2\})$ for a sufficiently large constant c. A similar result holds for the SSWM algorithm. Since the $(1+1)$ EA needs time $\Omega(n^{\ell_1})$ to cross the valley, here the Metropolis and SSWM algorithm are clearly superior.

In their time complexity analysis of the move acceptance hyper-heuristic (MAHH), Lissovoi, Oliveto, and Warwicker [LOW19] also consider the Metropolis algorithm. For the multimodal Cliff benchmark with constant cliff length d, they show that the Metropolis algorithm needs at least an expected number of $\tilde{\Omega}(n^{d-0.5})$ iterations to find the optimum, which is at most a little faster than the $\Theta(n^d)$ runtime of simple mutation-based algorithms, but is much worse than the $O(n \log n)$ performance of a self-adjusting $(1,\lambda)$ EA [FS21], the $O(n^3)$ runtime of the MAHH [LOW19], the $O(n^3 \log n)$ runtime of two self-adjusting fast artificial immune systems [COY21], and the $O(n^{3.9767})$ runtime of the $(1,\lambda)$ EA with the best static choice of λ [FS21] (improving the well-known $O(n^{25})$ runtime guarantee [JS07]). For the multimodal Jump benchmark, the Metropolis algorithm is said [LOW19] to have a runtime exponential in n with high probability regardless of the jump size m when the temperature is sufficiently small that the local optimum of the jump function is reached efficiently. This compares unfavorably with the known runtimes of many algorithms, which are all $\Theta(n^m)$ or better, see, e.g., [DJW02, DFK+16, DFK+18, DLMN17, HS18, Doe19b, Doe20a, RW20, ABD21].

In summary, we would not say that these results give a strong recommendation for using the Metropolis algorithm to optimize pseudo-Boolean functions. It excels on the VALLEY problem proposed in [OPH+18], however, this problem is essentially a one-dimensional problem and thus of a very different structure than most pseudo-Boolean problems. The possible $\tilde{O}(n^{0.5})$ runtime advantage over simple EAs on the CLIFF problem is interesting (if it exists, only a lower bound was shown in [LOW19]), but appears still small when recalling that the runtime of the simple EAs is $\Theta(n^d)$. Hence maybe most convincing result favoring the Metropolis algorithm is the quadratic runtime shown [JS98] for the random bipartition instances. Here, however, one also has to recall that random instances of NP-complete problems can by surprisingly easy. For example, the $(1+1)$ EA can solve random 3-SAT instances in the planted solution model in time $O(n \log n)$ when the clause density is at least logarithmic [DNS17]. These mildly positive results on the Metropolis algorithm have to be contrasted with several negative results, e.g., the exponential runtimes shown in [SH88, Weg05, Mee07]. Also, the parameter choice seems to be non-trivial. Whereas the best results in [OPH+18] are obtained by small values of α, recall that the runtime shown there is at least $n\alpha^{\Omega(d_1)}$ and the simple ONEMAX benchmark can only be optimized in polynomial time when α is as large as $\Omega(n \log n)$.

3.4.2 Non-Elitist Evolutionary Algorithms

The strategy to not stick with the best-so-far solution to enable the algorithm to leave a local optimum is also well-known in evolutionary computation. However, also there the rigorous support for such non-elitist algorithms is rather weak.

There is a fairly elaborate methodology to prove upper bounds on the runtime of non-elitist algorithms [Leh11, DL16, CDEL18, DK19], however, these tools so far could mostly be applied to settings where the evolutionary algorithm loses the best-so-far solution so rarely that it roughly imitates an elitist algorithm. A few analyses of particular algorithms point into a similar direction [JS07, RS14]. The existing general lower bound methods for non-elitist algorithms [Leh10, Doe21] in many cases allowed to prove that with just a little more non-elitism, the algorithm has no chance to optimize efficiently any pseudo-Boolean function with unique optimum.

As a concrete example, the standard (μ, λ) EA using mutation rate $\frac{1}{n}$ cannot optimize any pseudo-Boolean function with unique optimum in sub-exponential time if $\lambda \leq (1-\varepsilon)e\mu$, $\varepsilon > 0$ any constant. However, when $\lambda \geq (1+\varepsilon)e\mu$ and $\lambda = \omega(\log n)$, then the (μ, λ) EA optimizes ONEMAX and JUMP functions with constant jump size in essentially the same run-

time as the $(\mu + \lambda)$ EA. This suggests that there is at most a very small regime in which non-elitism can be successfully exploited. Such a small middle regime was shown to exist when optimizing CLIFF functions via the $(1, \lambda)$ EA. More specifically, Jägersküpper and Storch [JS07] showed that the $(1, \lambda)$ EA with $\lambda \geq 5 \ln n$ optimizes the CLIFF function with cliff length $n/3$ in time $\exp(5\lambda) \geq n^{25}$, whereas elitist mutation-based algorithms easily get stuck in the local optimum and then need at least $n^{n/3}$ expected time to optimize this CLIFF function. This result was very recently improved and runtime of $O(n^{3.9767})$ was shown for the asymptotically best choice of λ [FS21]. For the (μ, λ) EA optimizing jump functions, however, the existence of a profitable middle regime was disproven in [Doe20a]. Some more results exist that show situations where only an inefficient regime exists, e.g., when using (1+1)-type hillclimbers with fitness-proportionate selection to optimize linear pseudo-Boolean functions [HJKN08] or when using a mutation-only variant of the simple genetic algorithm to optimize ONEMAX [NOW09]. For the true simple genetic algorithm, such a result exists only when the population size is at most $\mu \leq n^{1/4-\varepsilon}$, but there is no proof that the situation improves with larger population size.

The few results showing that non-elitism can help to leave local optima include, besides the examples for the Metropolis algorithm discussed in Section 3.4.1 (see the last paragraph of that section for a summary) and the two "small middle regime" results discussed in the previous paragraph, the following. Dang, Eremeev, and Lehre [DEL21a] showed that a non-elitist EA with 3-tournament selection, population size $\lambda \geq c \log n$ for c a positive constant, and bitwise mutation with rate χ/n, $\chi = 1.09812$, can reach the optimum of the multi-modal FUNNEL problem with parameter $\omega \leq \frac{3}{4}n$ in expected time $O(n\lambda \log \lambda + n^2 \log n)$, whereas the $(\mu + \lambda)$ EA and the (μ, λ) EA cannot reach the optimum in time $2^{c'n}$, $c' > 0$ a suitable constant, with overwhelming probability. In [DEL21b], the same authors defined a class of functions having an exponential elitist $(\mu + \lambda)$ black-box complexity which can be solved in polynomial time by several non-elitist algorithms. Fajardo and Sudholt [FS21] showed that a self-adjusting variant of the $(1, \lambda)$ EA can optimize CLIFF functions in time $O(n \log n)$. Zheng, Zhang, Chen, and Yao [ZZCY21] proved that the $(1, \lambda)$ EA with offspring population size $\lambda = c \log_{\frac{e}{e-1}} n$ for the constant $c \geq 1$ can reach the global optimum of the time-linkage ONEMAX function in expected time $O(n^{3+c} \log n)$, while the $(1 + \lambda)$ EA with $\lambda \in [e^e, e^{n^{1/3}}]$ (as well as the $(1 + 1)$ EA analyzed in [ZCY21]) with $1 - o(1)$ probability cannot reach the global optimum in arbitrary long time.

More results exist for non-classical algorithms. In [PHST17], the strong-selection weak-mutation process from biology is regarded under an algorithmic perspective. It is shown that when setting the two parameters of this algorithm right, then it optimizes the $\textsc{Cliff}_d$ function, $d = \omega(\log n)$, in time $(n/\Omega(1))^d$, which is faster than the $(1+1)$ EA by a factor of $\exp(\Omega(d))$. It also optimizes the slightly artificial Balance function with high probability in time $O(n^{2.5})$, whereas the $(1+1)$ EA takes weakly exponential time. In [LOW19], a runtime analysis of the move-acceptance hyper-heuristic (MAHH) is performed. This can be seen as a variant of the Metropolis algorithm, where the probability of accepting an inferior solution is an algorithm parameter independent of the degree of inferiority. If this probability is chosen sufficiently small, then the MAHH optimizes all Cliff functions in time $O(n^3)$, significantly beating all other results on this function class. For the $\textsc{Jump}_k$ function, however, only a runtime guarantee of $O(n^{2k-1}/k)$ was proven. We note that aging as used in artificial immune systems can also lead to non-elitism and has been shown to speed-up leaving local optima, see, e.g., [OS14, COY20, COY21]. Since the concept of aging has not found many applications in heuristic search outside the small area of artificial immune systems used as optimizers, we omit a detailed discussion. Finally, we note that restarts obviously lead to algorithms that can be called non-elitist. We also discuss these not further since we view them, similar as parallel run or racing techniques, rather as mechanisms to be added to basic optimization heuristics. Overall, these examples prove that non-elitism can be helpful, but from their sparsity in the large amount of research on randomized search heuristics, their specificity, and their sometimes still high runtimes, we would not interpret them as strong indication for trying non-elitist approaches early.

3.5 A Lower Bound for the Unary Unbiased Black-Box Complexity

In this section we will show that the expected runtime of any unary unbiased black-box algorithm on the DLB problem is $\Omega(n^2)$, which together with the upper bound of Section 3.2 proves that the unary unbiased black-box complexity of the DLB problem is $\Theta(n^2)$. This result is very natural given that Lehre and Witt [LW12] have proven the same lower bound for the unary unbiased black-box complexity of the LO problem, which appears rather easier than the DLB problem.

To this end, we first prove the following lemma, which affirms that for a search point in a certain range of LO values, the probability of obtaining a search point with better fitness after one step is $O(\frac{1}{n})$.

Lemma 15. *Suppose that n is an even integer and that m is an integer such that $\frac{n}{4} < m < \frac{n}{3}$. Let x be a bit string and let $\ell := |\{j \in [1..2m+2] \mid x_j = 1|$ designate the number of ones in the leading $2m+2$ bit positions of x. We suppose that $1 \leq \ell \leq 2m+1$. Then for any unary unbiased operator V, we have $\Pr[\mathrm{LO}(V(x)) \geq 2m+2] \leq \frac{2}{n}$.*

Proof. We assume that for some $r \in [0..n]$, V is the unary unbiased operator that flips r bit positions chosen uniformly at random from the bit string. The conclusion for the general case then follows by applying the law of total probability and Lemma 2.

We furthermore assume that $2 + 2m - \ell \leq r \leq n - \ell$ since otherwise the claim is trivial. Indeed, $r < 2 + 2m - \ell$ implies $r + \ell < 2m + 2$. Since $r + \ell$ is an upper bound on the number of ones $V(x)$ has in the first $2m + 2$ positions, we obtain $\mathrm{LO}(V(x)) < 2m + 2$ with probability 1. If $r > n - \ell$, then at least one of the bits $x_j = 1$, $j \in [1..2m + 2]$, is flipped, also resulting in $\mathrm{LO}(V(x)) < 2m + 2$.

The event $\{\mathrm{LO}(V(x)) \geq 2m + 2\}$ happens if and only if all bits x_j such that $x_j = 1$ and $j \in [1..2m + 2]$, are not flipped, and all bits $x_j = 0$, $j \in [1..2m + 2]$, are flipped. Among all $\binom{n}{r}$ ways of flipping r bit positions in x, exactly $\binom{n-2m-2}{r-(2m+2-\ell)}$ of them flip all the 0s in the leading $2m + 2$ bit positions and leave all the 1s in the leading $2m + 2$ bit positions unchanged. Hence the probability $D_\ell(r)$ of the event $\{\mathrm{LO}(V(x)) \geq 2m + 2\}$ is given by

$$D_\ell(r) = \frac{\binom{n-2m-2}{r-(2m+2-\ell)}}{\binom{n}{r}} = \frac{(n-r)\cdots(n-r-\ell+1)\cdot r\cdots(r-2m+\ell-1)}{n\cdots(n-2m-1)}.$$

From our assumptions, with the variables n, m and r fixed, we have

$$\ell \in [\max\{1, 2m+2-r\}..\min\{2m+1, n-r\}]. \tag{9}$$

For $\ell \in [\max\{1, 2m+2-r\}..\min\{2m, n-r-1\}]$, we compute

$$\frac{D_{\ell+1}(r)}{D_\ell(r)} = \frac{2(\frac{n}{2}+m-\ell-r)+1}{r-2m+\ell-1} + 1. \tag{10}$$

We conclude that $D_{\ell+1}(r) > D_\ell(r)$ ($D_{\ell+1}(r) < D_\ell(r)$ resp.) is equivalent to $\frac{n}{2} + m \geq \ell + r$ ($\frac{n}{2} + m < \ell + r$ resp.). To proceed we distinguish three cases regarding the value of r.

Case $r \in [1..\frac{n}{2} - m - 1]$.

Since $\ell \leq 2m + 1$ (equation (9)), we have $\ell + r \leq 2m + 1 + \frac{n}{2} - m - 1 = \frac{n}{2} + m$, which implies $\frac{D_{\ell+1}(r)}{D_\ell(r)} > 1$ for all $\ell \in [1..2m + 1]$. It follows that $D_\ell(r) \leq D_{2m+1}(r)$ for all $\ell \leq 2m + 1$.

Case $r \in [\frac{n}{2} - m..\frac{n}{2} + m]$.

In this case $\frac{D_{\ell+1}(r)}{D_\ell(r)} > 1$ for $\ell \leq \frac{n}{2}+m-r$ and $\frac{D_{\ell+1}(r)}{D_\ell(r)} < 1$ for $\ell > \frac{n}{2}+m-r$, from which we conclude $D_\ell(r) \leq D_{\frac{n}{2}+m-r+1}(r)$.

Case $r \in [\frac{n}{2} + m + 1..n - 1]$.

In this case $\ell + r > r > \frac{n}{2} + m$, therefore $D_{\ell+1}(r) < D_\ell(r)$. Since ℓ is at least 1, we have $D_\ell(r) \leq D_1(r)$.

In short, we have established that

$$D_\ell(r) \leq \begin{cases} D_{2m+1}(r), & r \in [1..\frac{n}{2} - m - 1], \\ D_{\frac{n}{2}+m-r+1}(r), & r \in [\frac{n}{2} - m..\frac{n}{2} + m], \\ D_1(r), & r \in [\frac{n}{2} + m + 1..n - 1]. \end{cases}$$

By the definition of $D_\ell(r)$, we have

$$D_{2m+1}(r) = \binom{n-2m-2}{r-1}\binom{n}{r}^{-1},$$

$$D_{\frac{n}{2}+m-r+1}(r) = \frac{(n-r)\cdots(\frac{n}{2}-m)\cdot r\cdots(\frac{n}{2}-m)}{n\cdots(n-2m-1)},$$

$$D_1(r) = \frac{(n-r)\cdot r(r-1)\cdots(r-2m)}{n\cdots(n-2m-1)}.$$

We observe that $D_{2m+1}(r)$ is in fact the D_r that appeared in (3) of Lemma 8. Since we have proven $D_r \leq \frac{1}{n}$ for $r \geq 1$ in (4) of Lemma 8, the same estimate holds for $D_{2m+1}(r)$, i.e., $D_{2m+1}(r) \leq \frac{1}{n}$ for $r \geq 1$.

For $D_1(r)$, $r \in [\frac{n}{2} + m..n - 1]$, a straightforward calculation shows

$$\frac{D_1(r+1)}{D_1(r)} = \frac{2mn + n - 1 - (2m+2)r}{(r-2m)(n-r)} + 1.$$

Thus $D_1(r) < D_1(r+1)$ when $r < r^* := \frac{2mn+n-1}{2m+2}$ and $D_1(r) > D_1(r+1)$ when $r > r^*$. This implies

$$\begin{aligned} D_1(r) \leq D_1(\lceil r^* \rceil) &= \frac{n - \lceil r^* \rceil}{n}\frac{\lceil r^* \rceil}{n-1}\cdots\frac{\lceil r^* \rceil - 2m}{n-2m-1} \\ &< \frac{n - r^*}{n} = \frac{1}{2m+2}\left(1 + \frac{1}{n}\right) \leq \frac{1}{\frac{n}{2}+2}\left(1 + \frac{1}{n}\right) \leq \frac{2}{n}, \end{aligned}$$

where in the second last inequality we used the hypothesis $m > \frac{n}{4}$.

For $D_{\frac{n}{2}+m-r+1}(r)$, $r \in [\frac{n}{2} - m..\frac{n}{2} + m]$, we have

$$\frac{D_{\frac{n}{2}+m-r}(r+1)}{D_{\frac{n}{2}+m-r+1}(r)} = \frac{2r - n + 1}{n - r} + 1.$$

Hence $D_{\frac{n}{2}+m-r+1}(r)$ decreases with respect to r when $\frac{n}{2}-m \leq r \leq \frac{n}{2}-1$ and increases when $\frac{n}{2} \leq r \leq \frac{n}{2}+m$. Therefore we have for $r \in [\frac{n}{2}-m..\frac{n}{2}+m]$

$$D_{\frac{n}{2}+m-r+1}(r) \leq \max\{D_{2m+1}(\tfrac{n}{2}-m), D_1(\tfrac{n}{2}+m)\}.$$

Since it has already been shown that both $D_{2m+1}(r)$, $r \geq 1$, and $D_1(r)$, $r \in [\frac{n}{2}+m..n-1]$, are bounded by $\frac{2}{n}$, the same bound holds for $D_{\frac{n}{2}+m-r+1}(r)$, $r \in [\frac{n}{2}-m..\frac{n}{2}+m]$. We thus conclude

$$\Pr[\mathrm{LO}(V(x)) \geq 2m+2] = D_\ell(r) \leq \tfrac{2}{n}. \qquad \square$$

Let $\mathbf{1}$ denote $(1,\dots,1) \in \{0,1\}^n$. Following the standard notation, we write $\lceil i \rceil_2 := \{k \in 2\mathbb{Z} \mid k \geq i\}$ for all $i \in \mathbb{R}$. In the next lemma we will give a lower bound of $\Omega(n^2)$ on the unary unbiased black-box complexity of the DLB problem by considering the potential

$$h_t = \min\left\{2\left\lceil\frac{n}{3}\right\rceil, \max_{s\in[0..t]}\left\{\left\lceil\frac{n+1}{2}\right\rceil_2, \mathrm{HLB}_2(x^{(s)}), \mathrm{HLB}_2(\mathbf{1}-x^{(s)})\right\}\right\}, \quad (11)$$

that is, the maximum of the HLB_2 value of $x^{(s)}$, $s \in [0..t]$, and the HLB_2 value of $\mathbf{1}-x^{(s)}$, $s \in [0..t]$, capped into the interval $\left[\left\lceil\frac{n+1}{2}\right\rceil_2, 2\left\lceil\frac{n}{3}\right\rceil\right]$. The preceding lemma enables us to show that for any unary unbiased black-box algorithm, the expected gain in potential (11) at each iteration is $O(\frac{1}{n})$. The additive drift theorem will then imply the $\Omega(n^2)$ lower bound.

Lemma 16. *The unary unbiased black-box complexity of the* DLB *problem is* $\Omega(n^2)$.

Proof. Consider any unary unbiased black-box algorithm, let $x^{(t)}$ be the search point at time t. Let $T := \min\{t \in \mathbb{N} \mid \max\{\mathrm{HLB}_2(x^{(t)}), \mathrm{HLB}_2(\mathbf{1}-x^{(t)})\} \geq 2\lceil\frac{n}{3}\rceil\}$ be the first time the HLB_2 value of a search point or its complement is at least $2\lceil\frac{n}{3}\rceil$. We will use the additive drift theorem (Theorem 3) to prove $E[T] = \Omega(n^2)$, from which the claim follows since T is a trivial lower bound for the runtime.

Let $t < T$ and let $\mathcal{C}_m$ be the event $\{h_t = 2m\}$ for $m \in [\frac{1}{2}\lceil\frac{n+1}{2}\rceil_2..\lceil\frac{n}{3}\rceil - 1]$. By definition of h_t and T, one of the events $\mathcal{C}_m$ holds. We now estimate $E[h_{t+1}-h_t \mid \mathcal{C}_m]$. Let $x^{(s)}$, $s \in [0..t]$, be the individual chosen in iteration $t+1$ to generate $x^{(t+1)}$ (see line 4 of Algorithm 2, note that here we have $k=1$). Let $\ell := |\{i \in [1..2m+2] \mid x_i^{(s)} = 1\}|$ be the number of the ones in the leading $2m+2$ bit positions of $x^{(s)}$. Then $1 \leq \ell \leq 2m+1$ because otherwise h_t would be not less than $2m+2$.

If $h_{t+1} > h_t$, then either $x^{(t+1)}$ has no less than $2m+2$ leading ones or it has no less than $2m+2$ leading zeros, i.e.

$$\begin{aligned}&\{h_{t+1} > h_t\} \cap \mathcal{C}_m \\ &\quad = \big(\{\mathrm{HLB}_2(x^{(t+1)}) > 2m\} \cap \mathcal{C}_m\big) \dot\cup \big(\{\mathrm{HLB}_2(\mathbf{1} - x^{(t+1)}) > 2m\} \cap \mathcal{C}_m\big).\end{aligned}$$

Lemma 15 shows that the first scenario happens with probability at most $\frac{2}{n}$, that is,

$$\Pr[\mathrm{HLB}_2(x^{(t+1)}) > 2m \mid \mathcal{C}_m] \leq \tfrac{2}{n}.$$

We recall that $x^{(t+1)}$ is obtained from some $x^{(s)}$, $s \in [0..t]$, via a unary unbiased operator. By Lemma 5, $x_i^{(s)}$, $i \in [2m+3..n]$, are all independent and uniformly distributed in $\{0,1\}$ because of the condition $\mathcal{C}_m = \{h_t = 2m\}$. If we condition further on the event $\{\mathrm{HLB}_2(x^{(t+1)}) > 2m\} \cap \mathcal{C}_m$, then $x_i^{(t+1)}$, $i \in [2m+3..n]$, are all independent and uniformly distributed in $\{0,1\}$ and $x_i^{(t+1)} = 1$ for $i \in [1..2m+2]$. Hence Lemma 7 can be applied to $x^{(t+1)}$, yielding

$$\begin{aligned}&E[h_{t+1} - h_t \mid \mathrm{HLB}_2(x^{(t+1)}) > 2m, \mathcal{C}_m] \\ &\quad \leq E[\mathrm{HLB}_2(x^{(t+1)}) - 2m \mid \mathrm{HLB}_2(x^{(t+1)}) > 2m, \mathcal{C}_m] \\ &\quad \leq 2m + 4 - 2m = 4.\end{aligned}$$

For the second scenario, we consider $\mathbf{1} - x^{(s)}$ and $\mathbf{1} - x^{(t+1)}$ instead of $x^{(s)}$ and $x^{(t+1)}$ to obtain

$$\Pr[\mathrm{HLB}_2(\mathbf{1} - x^{(t+1)}) > 2m \mid \mathcal{C}_m] \leq \tfrac{2}{n}$$

and

$$E[h_{t+1} - h_t \mid \mathrm{HLB}_2(\mathbf{1} - x^{(t+1)}) > 2m, \mathcal{C}_m] \leq 4.$$

Combining the two scenarios, we have

$$\begin{aligned}E[h_{t+1} - h_t \mid \mathcal{C}_m] &= \Pr[h_{t+1} > h_t \mid \mathcal{C}_m] E[h_{t+1} - h_t \mid h_{t+1} > h_t, \mathcal{C}_m] \\ &\leq 2 \cdot \tfrac{2}{n} \cdot 4 = \tfrac{16}{n}.\end{aligned}$$

Now we examine the expected initial value of the potential. First we observe that $\Pr\big[\max\{\mathrm{HLB}_2(x^{(0)}), \mathrm{HLB}_2(\mathbf{1} - x^{(0)})\} \geq \frac{n}{2}\big] = O(2^{-\frac{n}{2}})$ since $x^{(0)}$ is generated uniformly at random in $\{0,1\}^n$. Thus

$$E[h_0] \leq \left\lceil \frac{n+1}{2} \right\rceil_2 + O(n 2^{-\frac{n}{2}}) = \frac{n}{2} + O(1).$$

The additive drift theorem (Theorem 3) thus implies

$$E[T] \geq \frac{2n/3 - n/2 - O(1)}{16/n} = \Theta(n^2). \qquad \square$$

Combining Lemma 11 with Lemma 16, we have proven the following theorem.

Theorem 17. *The unary unbiased black-box complexity of the* DLB *problem is* $\Theta(n^2)$.

4 Beyond Unary Unbiased Algorithms

We recall that in this work we are generally looking for unbiased algorithms as this is most natural when trying to solve a novel problem without much problem-specific understanding. In Sections 3 and 3.2, we have discussed unary unbiased algorithms. Our theory-guided approach has suggested the Metropolis algorithm, which with a $\Theta(n^2)$ runtime improved over the $O(n^3)$ runtime guarantee previously shown for various classic unbiased evolutionary algorithms. Our $\Omega(n^2)$ lower bound for all unary unbiased algorithms, however, also shows that further improvements are not possible with unary unbiased algorithms, and raises the question if algorithms of higher arity are more powerful for the DLB problem.

Following our theory-guided approach, we first exhibit in Section 4.1 that the binary unbiased black-box complexity of the DLB problem is $O(n \log n)$. We did not find a natural binary unbiased black-box algorithm for which we could show an $o(n^2)$ runtime, but by resorting to unrestricted arities, which allows for estimation-of-distribution algorithms, we detected that the *significance-based compact genetic algorithm* (sig-cGA) [DK20a] has a runtime of $O(n \log n)$ with high probability (Section 4.2).

4.1 Unbiased Black-Box Algorithms of Higher Arity

In this section we will exhibit a binary unbiased black-box algorithm (Algorithm 5) whose runtime on the DLB problem is $O(n \log n)$. Our algorithm builds on two ideas used in [DJK+11, Section 5] to design an $O(n \log n)$ binary unbiased black-box algorithm for the LO function: (i) We use a pair (x, y) of search points such that the bits i with $x_i = y_i$ are exactly the ones for which we know that the optimum also has this bit value in this bit position. (ii) We conduct a random binary search in $\{i \in [1..n] \mid x_i \neq y_i\}$ to find in logarithmic time a bit position such that flipping this bit in either x or y (which one will be visible from the fitness) sets it to the value the optimum has in this position. This operation increases the number of correctly detected bits by one. Different from the situation in [DJK+11], in our setting such a good bit value cannot always be detected from a fitness gain (due to the deceptive nature of the DLB problem). We overcome this by regarding

the HLB_1 function instead. Note that there is a (non-monotonic) one-to-one relation between the fitness values of DLB and HLB_1. Hence a black-box algorithm for DLB has access also to the HLB_1 value of any bit string.

We first recall from [DJK+11] the two binary unbiased variation operators randomWhereDifferent$(\cdot,\cdot)$ and switchIfDistanceOne$(\cdot,\cdot)$. The operator randomWhereDifferent$(\cdot,\cdot)$ takes two bit strings y and y' as input and outputs a bit string z of which the components z_i, $i \in [1..n]$, are determined by the following rule: we have $z_i = y_i$ if $y_i = y'_i$, and z_i is chosen uniformly at random from $\{0,1\}$ if $y_i \neq y'_i$. This operator is also (and better) known as uniform crossover, but to ease the comparison with [DJK+11] we keep the notation used there. The operator switchIfDistanceOne$(\cdot,\cdot)$ takes two bit strings y and y' as input and outputs y' if the Hamming distance between y and y' is one, otherwise it outputs y.

We now state our black-box algorithm (Algorithm 5). Since, as discussed, any black-box algorithm for the DLB problem also has access to the HLB_1 values, we use this function as well in the algorithm description. Algorithm 5 is initialized by generating a random search point x and its complement y (we note that taking the complement is the unary unbiased operator of flipping n bits).

In each iteration of the **for** loop, we ensure $\mathrm{HLB}_1(y) \leq \mathrm{HLB}_1(x)$ by exchanging x and y when necessary and run the subroutine in the lines 6-10, which flips one bit in y in a way that increases the HLB_1 value of y. To be more precise, the subroutine does a random binary search for such a bit position. At the beginning of the subroutine, y' is set to be x. In each iteration of the **repeat** loop, y'' is sampled from randomWhereDifferent(y, y'). If $\mathrm{HLB}_1(y'') > \mathrm{HLB}_1(y)$, then we accept y'' and attribute its value to y', otherwise we refuse y'' and proceed to the next iteration of the subroutine.

This subroutine terminates when y' and y differ only in one bit position. At this moment y is set to be equal to y'. Now x and y have exactly one more bit in common, which means the number of correctly detected bits is increased by one. By induction, after i iterations of the **for** loop, x and y agree on i bit positions. After n iterations, x and y are both equal to the optimum.

To analyze the time complexity of Algorithm 5, we first consider the runtime of the **repeat** subroutine.

Lemma 18. *The subroutine in lines 6-10 of Algorithm 5 finishes in an expected runtime of $O(\log n)$.*

Proof. For sake of simplicity we suppose $\mathrm{LO}(y) = 2m+1$ for some m at the beginning of the subroutine, the cases $\mathrm{DLB}(y) = 2m+1$ and $\mathrm{DLB}(y) = \mathrm{LO}(y) = 2m$ can be treated similarly.

Algorithm 5: A binary unbiased black-box algorithm for maximizing the DLB function.

```
1  initialization: generate a search point x uniformly at random in
     {0,1}^n and let y be the complement of x;
2  for i ∈ [1..n] do
3      if HLB_1(y) > HLB_1(x) then (x, y) ← (y, x);
4      y' ← x;
5      H ← HLB_1(y);
6      repeat
7          y'' ← randomWhereDifferent(y, y');
8          if HLB_1(y) < HLB_1(y'') then y' ← y'';
9          y ← switchIfDistanceOne(y, y');
10     until HLB(y) > H;
```

Since $\mathrm{HLB}_1(y'') > 2m+1 = \mathrm{HLB}_1(y)$ if and only if $y''_{2m+2} = 1$ and this second event happens with probability $\frac{1}{2}$ (line 7), the value of y'' is assigned to y' with probability $\frac{1}{2}$.

Now we consider the bit positions, other than the bit position $2m+2$, on which y and y' are different. Since the bit values of y'' on these positions are chosen at random, the probability that at least half of them coincide with corresponding bit values in y is at least $\frac{1}{2}$ by virtue of symmetry. Therefore, at the end of an iteration of the subroutine, with probability $\frac{1}{4}$ the number of bit positions other than the bit position $2m+2$, on which y and y' are different, is at most half as before. From this we conclude that in expected runtime $O(\log n)$, y and y' are only different in bit position $2m+2$. □

As discussed above, x and y are set to the optimum after having run the **for** loop for n times. Lemma 18 thus implies the following theorem on the binary unbiased black-box complexity of the DLB problem.

Theorem 19. *The binary unbiased black-box complexity of the* DLB *problem is* $O(n \log n)$.

Algorithm 5 reveals that the ability to learn how good solutions look like plays an important role in solving the DLB problem. This inspires the study of EDAs, and more specifically, the significance-based EDA proposed by Doerr and Krejca [DK20a] in the following.

4.2 Significance-Based Compact Genetic Algorithm (sig-cGA)

Estimation-of-distribution algorithms (EDAs) optimize a function $f : \{0,1\}^n \to \mathbb{R}$ by evolving a probabilistic model of the solution space $\{0,1\}^n$ in such a direction that this probabilistic model becomes more likely to sample the optimum.

We now present the *sig-cGA* (Algorithm 6) proposed by Doerr and Krejca [DK20a]. To this end, we first define of the notion of *frequency vectors*. A frequency vector $\tau = (\tau_i)_{i\in[1..n]}$ is a vector whose components represent a probability, i.e., $\tau_i \in [0,1]$, $i \in [1..n]$. A probability distribution on $\{0,1\}^n$ is associated to a frequency vector τ in the following way: an individual $x \in \{0,1\}^n$ follows this probability distribution if and only if each component x_i of x independently follows the Bernoulli distribution of parameter τ_i.

The sig-cGA utilizes frequency vectors whose components take value in $\{\frac{1}{n}, \frac{1}{2}, 1-\frac{1}{n}\}$ to represent a probability distribution on $\{0,1\}^n$ in the above way. For each bit position $i \in [1..n]$, the sig-cGA keeps a history $H_i \in \{0,1\}^*$ for significance inferring that will be explained in detail in the following. In iteration t, two individuals, x and y, are independently sampled from the probability distribution associated with $\tau^{(t)}$. They are then evaluated under f, the one with better fitness is called the winner and is denoted by z. In the case of a tie the winner is chosen at random. Now for each $i \in [1..n]$, the value of z_i is added to the history H_i. If a statistical significance of 1 (0 resp.) is detected in H_i, then $\tau_i^{(t+1)}$ is set to $1-\frac{1}{n}$ ($\frac{1}{n}$ resp.) and the history H_i is emptied. To be more precise, we define in the following the function sig_ε taking values in $\{\textsc{up}, \textsc{down}, \textsc{stay}\}$ and we say that a significance of 1 (0 resp.) is detected when $\mathrm{sig}_\varepsilon\left(\tau_i^{(t)}, H_i\right) = \textsc{up}$ ($\textsc{down}$ resp.).

For all $\varepsilon, \mu \in \mathbb{R}^+$, let $s(\varepsilon,\mu) = \varepsilon \max\{\sqrt{\mu \log n}, \log n\}$. For all $H \in \{0,1\}^*$, let $H[k]$ be the string of its last k bits and let $\|H[k]\|_0$ ($\|H[k]\|_1$ resp.) denote the number of zeros (resp. ones) in H. Then for all $p \in \{\frac{1}{n}, \frac{1}{2}, 1-\frac{1}{n}\}$ and $H \in \{0,1\}^*$, $\mathrm{sig}_\varepsilon(p,H)$ is defined by

$$\mathrm{sig}_\varepsilon(p,H) = \begin{cases} \textsc{up} & \text{if } p \in \left\{\frac{1}{n}, \frac{1}{2}\right\} \wedge \exists m \in \mathbb{N}: \\ & \|H[2^m]\|_1 \geq 2^m p + s(\varepsilon, 2^m p), \\ \textsc{down} & \text{if } p \in \left\{\frac{1}{2}, 1-\frac{1}{n}\right\} \wedge \exists m \in \mathbb{N}: \\ & \|H[2^m]\|_0 \geq 2^m (1-p) + s(\varepsilon, 2^m(1-p)), \\ \textsc{stay} & \text{else.} \end{cases}$$

The following lemma (Lemma 2 in [DK20a]) shows that the sig-cGA, with probability at least $1 - n^{-\varepsilon/3}\log_2 k$, does not detect a significance at a

Algorithm 6: The sig-cGA with parameter ε and significance function sig_ε optimizing $f : \{0,1\}^n \to \mathbb{R}$

$t \leftarrow 0$;
for $i \in [1..n]$ **do** $\tau_i^{(0)} \leftarrow \frac{1}{2}$ and $H_i \leftarrow \emptyset$;
repeat
  $x, y \leftarrow$ offspring sampled with respect to $\tau^{(t)}$;
  $z \leftarrow$ winner of x and y with respect to f, chosen at random in case of a tie;
  for $i \in [1..n]$ **do**
    $H_i \leftarrow H_i \circ z_i$;
    if $\mathrm{sig}_\varepsilon\left(\tau_i^{(t)}, H_i\right) = \textsc{up}$ **then** $\tau_i^{(t+1)} \leftarrow 1 - \frac{1}{n}$;
    else if $\mathrm{sig}_\varepsilon\left(\tau_i^{(t)}, H_i\right) = \textsc{down}$ **then** $\tau_i^{(t+1)} \leftarrow \frac{1}{n}$;
    else $\tau_i^{(t+1)} \leftarrow \tau_i^{(t)}$;
    if $\tau_i^{(t+1)} \neq \tau_i^{(t)}$ **then** $H_i \leftarrow \emptyset$;
    $t \leftarrow t + 1$;
until *termination criterion met*;

position with no bias in selection, that is, with a high probability it does not detect a false significance.

Lemma 20 (Lemma 2 in [DK20a])**.** *Consider the sig-cGA with $\varepsilon \geq 1$. Let $i \in [1..n]$ be a bit position and suppose that the distribution of 1s in H_i follows a binomial law with k tries and success probability τ_i. Then $\Pr\left[\mathrm{sig}_\varepsilon\left(\tau_i^{(t)}, H_i\right) \neq \textsc{stay}\right] \leq n^{-\varepsilon/3} \log_2 k$, that is, τ_i changes with a probability of at most $n^{-\varepsilon/3} \log_2 k$.*

The preceding lemma readily implies the following corollary.

Corollary 21. *Consider the sig-cGA with $\varepsilon \geq 1$. Let $i \in [1..n]$ be a bit position and suppose that the distribution of 1s in H_i follows a binomial law with k tries and a success probability of at least τ_i. Then τ_i decreases in an iteration with a probability of at most $n^{-\varepsilon/3} \log_2 k$.*

4.2.1 The Sig-cGA Solves the DLB Problem in $O(n \log n)$ Time

Now we show that with probability at least $1 - O\big(n^{2-\varepsilon/3} \log^2 n\big)$, the sig-cGA samples the optimum of the DLB function within $O(n \log n)$ fitness evaluations (Theorem 25). First we show that when no τ_i is set to $\frac{1}{n}$, the

probability of adding a 1 to the history H_i in an iteration is at least τ_i, which will allow us to use Corollary 21.

Lemma 22. *Let $m \in [1..\frac{n}{2}-1]$. Let $(\tau_i)_{i\in[1..n]} \in \{\frac{1}{n}, \frac{1}{2}, 1-\frac{1}{n}\}^n$ be a frequency vector. Consider one iteration of the sig-cGA optimizing the* DLB *function. Let x and y be the two individuals sampled according to $(\tau_i)_{i\in[1..n]}$. Then, conditioning on $\{\min\{\mathrm{DLB}(x), \mathrm{DLB}(y)\} \geq 2m\}$, a 1 is saved in H_{2m+1} with probability*

$$\begin{aligned} &p_{(\tau_{2m+1},\tau_{2m+2})} \\ &\qquad := \tau_{2m+1}\big(-\tau_{2m+2}^2 + 3\tau_{2m+2} + (\tau_{2m+2}^2 - 3\tau_{2m+2} + 1)\tau_{2m+1}\big). \end{aligned} \tag{12}$$

If $\tau_{2m+1}, \tau_{2m+2} \in \{\frac{1}{2}, 1-\frac{1}{n}\}$, then the above term is bounded from below by τ_{2m+1}. If further $\tau_{2m+1} = \frac{1}{2}$ and $\tau_{2m+2} \in \{\frac{1}{2}, 1-\frac{1}{n}\}$, then the above term is bounded from below by $\frac{9}{16}$.

Proof. Under the condition $\{\min\{\mathrm{DLB}(x), \mathrm{DLB}(y)\} \geq 2m\}$, the bit value saved in H_{2m+1} is completely determined by the bits x_{2m+1}, x_{2m+2}, y_{2m+1}, and y_{2m+2}. In the following we assume $m = 0$ for the ease of presentation, but the general result can be obtained in exactly the same way. We calculate

$$\begin{aligned} &\Pr[1 \text{ is saved in } H_1] \\ &= \Pr[x_1 = x_2 = 1] \\ &\quad + \Pr[x_1 = 1, x_2 = 0]\big(\Pr[y_1 = 1] + \tfrac{1}{2}\Pr[y_1 = 0, y_2 = 1]\big) \\ &\quad + \Pr[x_1 = 0, x_2 = 1]\big(\Pr[y_1 = y_2 = 1] + \tfrac{1}{2}\Pr[y_1 = 1, y_2 = 0]\big) \\ &\quad + \Pr[x_1 = 0, x_2 = 0]\Pr[y_1 = y_2 = 1] \\ &= \tau_1\tau_2 + \tau_1(1-\tau_2)\big(\tau_1 + \tfrac{1}{2}(1-\tau_1)\tau_2\big) \\ &\quad + (1-\tau_1)\tau_2\big(\tau_1\tau_2 + \tfrac{1}{2}\tau_1(1-\tau_2)\big) \\ &\quad + (1-\tau_1)(1-\tau_2)\tau_1\tau_2 \\ &= \tau_1\big(-\tau_2^2 + 3\tau_2 + \big(\tau_2^2 - 3\tau_2 + 1\big)\tau_1\big). \end{aligned}$$

Now if $\tau_1, \tau_2 \in \{\frac{1}{2}, 1-\frac{1}{n}\}$, then

$$-\tau_2^2 + 3\tau_2 = -(\tau_2 - \tfrac{3}{2})^2 + \tfrac{9}{4} \geq -(\tfrac{1}{2} - \tfrac{3}{2})^2 + \tfrac{9}{4} = \tfrac{5}{4}.$$

Thus we have $\tau_2^2 - 3\tau_2 + 1 \leq -\frac{1}{4} < 0$ and

$$\begin{aligned} \frac{p_{(\tau_1,\tau_2)}}{\tau_1} &= -\tau_2^2 + 3\tau_2 + (\tau_2^2 - 3\tau_2 + 1)\tau_1 \\ &\geq -\tau_2^2 + 3\tau_2 + (\tau_2^2 - 3\tau_2 + 1)\big(1 - \tfrac{1}{n}\big) \\ &= -\tfrac{1}{n}(\tau_2^2 - 3\tau_2) + 1 - \tfrac{1}{n} \\ &\geq -\tfrac{5}{4n} + 1 - \tfrac{1}{n} = 1 + \tfrac{1}{4n} > 1. \end{aligned}$$

Supposing further that $\tau_1 = \frac{1}{2}$, we have

$$\frac{p(\tau_1, \tau_2)}{\tau_1} = \tfrac{1}{2}(-\tau_2^2 + 3\tau_2 + 1) \geq \tfrac{1}{2}(\tfrac{5}{4} + 1) = \tfrac{9}{8}. \qquad \square$$

Corollary 23. *If in an iteration of the sig-cGA, the frequency vector is in $\{\frac{1}{2}, 1 - \frac{1}{n}\}^n$, then for any $i \in [1..n]$, a 1 is saved in H_i with probability at least τ_i.*

Proof. Let x and y be the two individuals sampled in this iteration. Let $i = 2m + 1$ for some $m \in [1..\frac{n}{2} - 1]$. Then the preceding lemma shows that conditioning on $\{\min\{\mathrm{DLB}(x), \mathrm{DLB}(y)\} \geq 2m\}$, a 1 is saved in H_{2m+1} with probability at least τ_{2m+1}. Under the condition $\{\min\{\mathrm{DLB}(x), \mathrm{DLB}(y)\} < 2m\}$, the probability that a 1 is saved in H_{2m+1} is equal to the probability τ_{2m+1} that a 1 is sampled in this position, because bit $2m + 1$ is not relevant for the selection. Combining the two cases, the claim follows for $i = 2m + 1$. The symmetry between $i = 2m + 1$ and $i = 2m + 2$ concludes the proof. $\square$

Lemma 24. *Consider the sig-cGA with $\varepsilon > 3$. The probability that during the first k iterations at least one frequency decreases is at most $kn^{1-\varepsilon/3} \log_2 k$.*

Proof. Consider the event that in the first t iterations, no frequency has ever been decreased. Denote its probability by $p^{(t)}$.

Conditioning on this event, Corollary 23 can be applied to verify the hypothesis of Corollary 21, which implies that the conditional probability that no frequency decreases in the $(t + 1)$-th iteration is at least $1 - n^{1-\varepsilon/3} \log_2 k$. Therefore $p^{(t+1)} \geq p^{(t)}(1 - n^{1-\varepsilon/3} \log_2 k)$. By induction, $p^{(k)} \geq (1 - n^{1-\varepsilon/3} \log_2 k)^k \geq 1 - kn^{1-\varepsilon/3} \log_2 k$. $\square$

Theorem 25. *The runtime of the sig-cGA with $\varepsilon > 6$ on* DLB *is $O(n \log n)$ with probability at least $1 - O\big(n^{2-\varepsilon/3} \log^2 n\big)$.*

Proof. By taking $k = O(n \log n)$ in Lemma 24, we obtain that with probability at least $1 - O\big(n^{2-\varepsilon/3} \log^2 n\big)$, no frequency decreases in the first $O(n \log n)$ iterations. We condition on this event in what follows.

During the runtime of the sig-cGA, a block (x_{2m+1}, x_{2m+2}) is called *critical* if m is such that $\tau_i = 1 - \frac{1}{n}$ for all $i \in [1..2m]$, and that $\tau_{2m+1} = \frac{1}{2}$ or $\tau_{2m+2} = \frac{1}{2}$. Suppose that such a critical block is created with $\tau_{2m+1} = \frac{1}{2}$. We prove that the history of position $2m + 1$ saves 1s significantly more often than 0s and hence the frequency τ_{2m+1} is set to $1 - \frac{1}{n}$ after $O(\log n)$ iterations.

Let O denote the event that we save a 1 in H_{2m+1} in one iteration. We now calculate a lower bound of O under the condition that no frequency is decreased within first $O(n \log n)$ iterations.

Let A denote the event that at least one of the two individuals sampled in an iteration has a DLB value smaller than $2m$. When A happens, a 1 is saved in H_{2m+1} with probability $\tau_{2m+1} = \frac{1}{2}$. If A does not happen, then the probability of saving a 1 to H_{2m+1} is equal to $p(\tau_{2m+1}, \tau_{2m+2})$ defined in (12). Hence we can decompose $\Pr[O]$ as

$$\Pr[O] = \Pr[A]\tau_{2m+1} + \Pr[\bar{A}]p(\tau_{2m+1}, \tau_{2m+2}).$$

Since $\bar{A}$ is equivalent to both individuals have 1s in the first $2m$ bit positions, we have $\Pr[\bar{A}] = \left(\left(1 - \frac{1}{n}\right)^{2m}\right)^2 \geq \left(\left(1 - \frac{1}{n}\right)^{n-2}\right)^2 \geq (e^{-1})^2 = e^{-2}$. With Lemma 22, this implies

$$\begin{aligned}\Pr[O] \geq& (1 - \Pr[\bar{A}])\tau_{2m+1} + \Pr[\bar{A}]\tfrac{9}{8}\tau_{2m+1}\\ \geq& \big(1 + \tfrac{1}{8}\Pr[\bar{A}]\big)\tau_{2m+1} = (1 + \tfrac{1}{8}e^{-2})\tau_{2m+1},\end{aligned}$$

since we assumed that $\tau_{2m+1} = \frac{1}{2}$.

With this lower bound on $\Pr[O]$ we bound the probability that a significance of 1s in H_{2m+1} is detected within $k = O(\log n)$ iterations. To this end, we consider the process $X \sim \mathrm{Bin}(k, (1 + e^{-2}/8)\tau_{2m+1})$, which is stochastically dominated by the actual process of saving 1s at position H_{2m+1}. It follows from the definition that

$$\begin{aligned}&\Pr[X \leq k\tau_{2m+1} + s(\varepsilon, k\tau_{2m+1})]\\ &\quad= \Pr\big[X \leq E[X] - \big(\tfrac{k}{8}e^{-2}\tau_{2m+1} - s(\varepsilon, k\tau_{2m+1})\big)\big].\end{aligned}$$

For $k > 64e^4\varepsilon^2\tau_{2m+1}^{-1}\log n = \Theta(\log n)$ we have $s(\varepsilon, k\tau_{2m+1}) = \varepsilon\sqrt{k\tau_{2m+1}\log n}$ and thus $\frac{k}{8}e^{-2}\tau_{2m+1} - s(\varepsilon, k\tau_{2m+1}) > 0$. Let $c := 64e^4\varepsilon^2\tau_{2m+1}^{-1}$ and consider $k > 4c\log n$ iterations. We have

$$s(\varepsilon, k\tau_{2m+1}) = \varepsilon\sqrt{k\tau_{2m+1}\log n} < \varepsilon k\sqrt{\frac{\tau_{2m+1}}{4c}} = \frac{\tau_{2m+1}k}{16e^2},$$

which implies

$$\frac{k}{8}e^{-2}\tau_{2m+1} - s(\varepsilon, k\tau_{2m+1}) > \frac{\tau_{2m+1}k}{16e^2} =: \lambda,$$

and hence

$$\Pr\big[X \leq E[X] - \big(\tfrac{k}{8}e^{-2}\tau_{2m+1} - s(\varepsilon, k\tau_{2m+1})\big)\big] \leq \Pr[X \leq E[X] - \lambda].$$

We use a Chernoff inequality to calculate an upper bound for this probability. By definition $\mathrm{Var}[X] = k(1 + e^{-2}/8)\tau_{2m+1}(1 - (1 + e^{-2}/8)\tau_{2m+1}) \geq \lambda$. It is straightforward that

$$\frac{\lambda^2}{\mathrm{Var}[X]} > \frac{\lambda^2}{k\tau_{2m+1}} = \frac{k\tau_{2m+1}}{256e^4} > \frac{c\tau_{2m+1}\log n}{64e^4} = \varepsilon^2\log n.$$

Now the Chernoff inequality (Theorem 4) implies

$$\Pr[X \leq E[X] - \lambda] \leq \exp\left(\frac{-\lambda^2}{3\mathrm{Var}[X]}\right) < \exp\left(-\frac{\varepsilon^2}{3}\log n\right) = n^{-\frac{\varepsilon^2}{3}}.$$

We have thus proven that with probability at most $n^{-\frac{\varepsilon^2}{3}}$, τ_{2m+1} is not set to $1-\frac{1}{n}$ after $O(k) = O(4c\log n) = O(\log n)$ iterations. Due to the symmetry between positions $2m+1$ and $2m+2$, the same holds true for bit positions $2m+2$, $m \in [1..\frac{n}{2}-1]$. Hence with probability at most $n^{1-\frac{\varepsilon^2}{3}}$, for some $i \in [1..n]$, τ_i is not set to $1-\frac{1}{n}$ within the first $O(n\log n)$ iterations, that is, with probability at least $1-n^{1-\frac{\varepsilon^2}{3}}$, every τ_i, $i \in [1..n]$, is set to $1-\frac{1}{n}$ within the first $O(nk) = O(n\log n)$ iterations.

Once this is achieved, the probability to sample the optimum $\mathbf{1} = (1,\dots,1)$ from $(\tau_i)_{i\in[1..n]}$ is equal to $\left(1-\frac{1}{n}\right)^n > \frac{1}{2e}$. Let $\varepsilon' > \frac{\varepsilon}{3} - 2$ be a constant. Since we condition on no frequency dropping, the sig-cGA samples $\mathbf{1}$ at least once within $\varepsilon'\frac{1}{2}\left(\log\left(\frac{2e}{2e-1}\right)\right)^{-1}\log n = O(\log n)$ iterations with a probability of at least $1-\left(1-\frac{1}{2e}\right)^{\varepsilon'\left(\log\left(\frac{2e}{2e-1}\right)\right)^{-1}\log n} = 1-n^{-\varepsilon'}$.

Recall that no frequency decreases within the first $O(n\log n)$ iterations with probability at least $1-O\left(n^{2-\varepsilon/3}\log^2 n\right)$. Combining the preceding results, we have proven that the optimum is sampled within $O(n\log n)$ iterations with probability at least

$$\left(1-O\left(n^{2-\varepsilon/3}\log^2 n\right)\right)\left(1-n^{1-\frac{\varepsilon^2}{3}}\right)(1-n^{-\varepsilon'}) = 1-O\left(n^{2-\varepsilon/3}\log^2 n\right)$$

because of the definition of ε' and the inequality $2-\frac{\varepsilon}{3} > 1-\frac{\varepsilon^2}{3}$. □

5 Experiments

To see how the algorithms compare on concrete problem sizes, we ran the $(1+1)$ EA, UMDA, Metropolis algorithm, and sig-cGA on the DLB function for $n = 40, 80, \dots, 200$. We used the standard mutation rate $p = 1/n$ for the $(1+1)$ EA, the population sizes $\mu = 3n\ln n$ and $\lambda = 12\mu$ for the UMDA (as in [DK20b]), and the temperature parameter $\alpha = 3$ (greater than $\sqrt{2}+1$ as suggested from Theorem 14) for the Metropolis algorithm. For the sig-cGA, we took $\epsilon = 2.5$ since we observed that this was enough to prevent frequencies from moving to an unwanted value, which only happened one time for $n = 40$. Being still very slow, for this algorithm we could only perform 10 runs for problem sizes 40, 80, and 120, and 7 runs for problem size 160.

Our experiments clearly show an excellent performance of the Metropolis algorithm, which was suggested as interesting algorithm by our theoretical

analysis. The two EDAs, however, perform worse than what the asymptotic results suggest. Such a discrepancy between theoretical predictions and experimental results, stemming from the details hidden in the asymptotic notation, has been observed before and is what triggered the research area of *algorithm engineering* [MHS01].

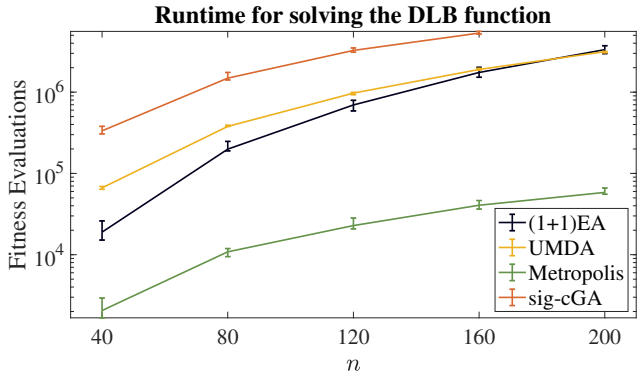

Figure 1: The median number of fitness evaluations (with the first and third quartiles) of $(1+1)$ EA, UMDA, Metropolis algorithm, and the sig-cGA on DLB with $n \in \{40, 80, 120, 160, 200\}$ in 20 independent runs (10 runs for $n \in \{40, 80, 120\}$ and 7 runs for $n = 160$ for the sig-cGA).

6 Conclusion and Outlook

To help choosing an efficient randomized search heuristic when faced with a novel problem, we proposed a theory-guided approach based on black-box complexity arguments and applied it to the recently proposed DLB function. Our approach suggested the Metropolis algorithm, for which little theoretical support before existed. Both a mathematical runtime analysis and our experiments proved it to be significantly superior to all previously analyzed algorithms for the DLB problem.

We believe that our approach, in principle and in a less rigorous way, can also be followed by researchers and practitioners outside the theory community. Our basic approach of (i) trying to understand how the theoretically best-possible algorithm for a given problem could look like and then (ii) using this artificial and problem-specific algorithm as indicator for promising established search heuristics, can also be followed by experimental methods and by non-rigorous intuitive considerations.

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.

This work was also supported by Guangdong Basic and Applied Basic Research Foundation (Grant No. 2019A1515110177), Guangdong Provincial Key Laboratory (Grant No. 2020B121201001), the Program for Guangdong Introducing Innovative and Enterpreneurial Teams (Grant No. 2017ZT07X386), Shenzhen Science and Technology Program (Grant No. KQTD2016112514355531).

References

[AAD+19] Peyman Afshani, Manindra Agrawal, Benjamin Doerr, Carola Doerr, Kasper Green Larsen, and Kurt Mehlhorn. The query complexity of a permutation-based variant of Mastermind. *Discrete Applied Mathematics*, 260:28–50, 2019.

[ABD20a] Denis Antipov, Maxim Buzdalov, and Benjamin Doerr. Fast mutation in crossover-based algorithms. In *Genetic and Evolutionary Computation Conference, GECCO 2020*, pages 1268–1276. ACM, 2020.

[ABD20b] Denis Antipov, Maxim Buzdalov, and Benjamin Doerr. First steps towards a runtime analysis when starting with a good solution. In *Parallel Problem Solving From Nature, PPSN 2020, Part II*, pages 560–573. Springer, 2020.

[ABD21] Denis Antipov, Maxim Buzdalov, and Benjamin Doerr. Lazy parameter tuning and control: choosing all parameters randomly from a power-law distribution. In *Genetic and Evolutionary Computation Conference, GECCO 2021*, pages 1115–1123. ACM, 2021.

[AD11] Anne Auger and Benjamin Doerr, editors. *Theory of Randomized Search Heuristics*. World Scientific Publishing, 2011.

[AD20] Denis Antipov and Benjamin Doerr. Runtime analysis of a heavy-tailed $(1 + (\lambda, \lambda))$ genetic algorithm on jump functions. In *Parallel Problem Solving From Nature, PPSN 2020, Part II*, pages 545–559. Springer, 2020.

[AD21] Denis Antipov and Benjamin Doerr. A tight runtime analysis for the $(\mu + \lambda)$ EA. *Algorithmica*, 83:1054–1095, 2021.

[ADK19] Denis Antipov, Benjamin Doerr, and Vitalii Karavaev. A tight runtime analysis for the $(1 + (\lambda, \lambda))$ GA on LeadingOnes. In *Foundations of Genetic Algorithms, FOGA 2019*, pages 169–182. ACM, 2019.

[ADK20] Denis Antipov, Benjamin Doerr, and Vitalii Karavaev. The $(1 + (\lambda, \lambda))$ GA is even faster on multimodal problems. In *Genetic and Evolutionary Computation Conference, GECCO 2020*, pages 1259–1267. ACM, 2020.

[AW09] Gautham Anil and R. Paul Wiegand. Black-box search by elimination of fitness functions. In *Foundations of Genetic Algorithms, FOGA 2009*, pages 67–78. ACM, 2009.

[BD17] Maxim Buzdalov and Benjamin Doerr. Runtime analysis of the $(1 + (\lambda, \lambda))$ genetic algorithm on random satisfiable 3-CNF formulas. In *Genetic and Evolutionary Computation Conference, GECCO 2017*, pages 1343–1350. ACM, 2017.

[BDN10] Süntje Böttcher, Benjamin Doerr, and Frank Neumann. Optimal fixed and adaptive mutation rates for the LeadingOnes problem. In *Parallel Problem Solving from Nature, PPSN 2010*, pages 1–10. Springer, 2010.

[BLS14] Golnaz Badkobeh, Per Kristian Lehre, and Dirk Sudholt. Unbiased black-box complexity of parallel search. In *Parallel Problem Solving from Nature, PPSN 2014*, pages 892–901. Springer, 2014.

[CDEL18] Dogan Corus, Duc-Cuong Dang, Anton V. Eremeev, and Per Kristian Lehre. Level-based analysis of genetic algorithms and other search processes. *IEEE Transactions on Evolutionary Computation*, 22:707–719, 2018.

[Che52] Herman Chernoff. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *Annals of Mathematical Statistics*, 23:493–507, 1952.

[COY20] Dogan Corus, Pietro S. Oliveto, and Donya Yazdani. When hypermutations and ageing enable artificial immune systems to outperform evolutionary algorithms. *Theoretical Computer Science*, 832:166–185, 2020.

[COY21] Dogan Corus, Pietro S. Oliveto, and Donya Yazdani. Automatic adaptation of hypermutation rates for multimodal optimisation. In *Foundations of Genetic Algorithms, FOGA 2021*, pages 4:1–4:12. ACM, 2021.

[DD18] Benjamin Doerr and Carola Doerr. Optimal static and self-adjusting parameter choices for the $(1 + (\lambda, \lambda))$ genetic algorithm. *Algorithmica*, 80:1658–1709, 2018.

[DDE15] Benjamin Doerr, Carola Doerr, and Franziska Ebel. From black-box complexity to designing new genetic algorithms. *Theoretical Computer Science*, 567:87–104, 2015.

[DDY20] Benjamin Doerr, Carola Doerr, and Jing Yang. Optimal parameter choices via precise black-box analysis. *Theoretical Computer Science*, 801:1–34, 2020.

[DEL21a] Duc-Cuong Dang, Anton V. Eremeev, and Per Kristian Lehre. Escaping local optima with non-elitist evolutionary algorithms. In *AAAI Conference on Artificial Intelligence, AAAI 2021*, pages 12275–12283. AAAI Press, 2021.

[DEL21b] Duc-Cuong Dang, Anton V. Eremeev, and Per Kristian Lehre. Non-elitist evolutionary algorithms excel in fitness landscapes with sparse deceptive regions and dense valleys. In *Genetic and Evolutionary Computation Conference, GECCO 2021*, pages 1133–1141. ACM, 2021.

[DFK+16] Duc-Cuong Dang, Tobias Friedrich, Timo Kötzing, Martin S. Krejca, Per Kristian Lehre, Pietro S. Oliveto, Dirk Sudholt, and Andrew M. Sutton. Escaping local optima with diversity mechanisms and crossover. In *Genetic and Evolutionary Computation Conference, GECCO 2016*, pages 645–652. ACM, 2016.

[DFK+18] Duc-Cuong Dang, Tobias Friedrich, Timo Kötzing, Martin S. Krejca, Per Kristian Lehre, Pietro S. Oliveto, Dirk Sudholt, and Andrew M. Sutton. Escaping local optima using crossover with emergent diversity. *IEEE Transactions on Evolutionary Computation*, 22:484–497, 2018.

[DGWY19] Benjamin Doerr, Christian Gießen, Carsten Witt, and Jing Yang. The $(1 + \lambda)$ evolutionary algorithm with self-adjusting mutation rate. *Algorithmica*, 81:593–631, 2019.

[DJK+11] Benjamin Doerr, Daniel Johannsen, Timo Kötzing, Per Kristian Lehre, Markus Wagner, and Carola Winzen. Faster black-box algorithms through higher arity operators. In *Foundations of Genetic Algorithms, FOGA 2011*, pages 163–172. ACM, 2011.

[DJTW02] Stefan Droste, Thomas Jansen, Karsten Tinnefeld, and Ingo Wegener. A new framework for the valuation of algorithms for black-box optimization. In *Foundations of Genetic Algorithms, FOGA 2002*, pages 253–270. Morgan Kaufmann, 2002.

[DJW02] Stefan Droste, Thomas Jansen, and Ingo Wegener. On the analysis of the (1+1) evolutionary algorithm. *Theoretical Computer Science*, 276:51–81, 2002.

[DJW06] Stefan Droste, Thomas Jansen, and Ingo Wegener. Upper and lower bounds for randomized search heuristics in black-box optimization. *Theory of Computing Systems*, 39:525–544, 2006.

[DK19] Benjamin Doerr and Timo Kötzing. Multiplicative up-drift. In *Genetic and Evolutionary Computation Conference, GECCO 2019*, pages 1470–1478. ACM, 2019.

[DK20a] Benjamin Doerr and Martin S. Krejca. Significance-based estimation-of-distribution algorithms. *IEEE Transactions on Evolutionary Computation*, 24:1025–1034, 2020.

[DK20b] Benjamin Doerr and Martin S. Krejca. The univariate marginal distribution algorithm copes well with deception and epistasis. In *Evolutionary Computation in Combinatorial Optimization, EvoCOP 2020*, pages 51–66. Springer, 2020.

[DK20c] Benjamin Doerr and Martin S. Krejca. The univariate marginal distribution algorithm copes well with deception and epistasis. *CoRR*, abs/2007.08277, 2020.

[DKLW13] Benjamin Doerr, Timo Kötzing, Johannes Lengler, and Carola Winzen. Black-box complexities of combinatorial problems. *Theoretical Computer Science*, 471:84–106, 2013.

[DL16] Duc-Cuong Dang and Per Kristian Lehre. Runtime analysis of non-elitist populations: from classical optimisation to partial information. *Algorithmica*, 75:428–461, 2016.

[DL17] Carola Doerr and Johannes Lengler. Introducing elitist black-box models: When does elitist behavior weaken the performance of evolutionary algorithms? *Evolutionary Computation*, 25, 2017.

[DL18] Carola Doerr and Johannes Lengler. The (1+1) elitist black-box complexity of LeadingOnes. *Algorithmica*, 80:1579–1603, 2018.

[DLMN17] Benjamin Doerr, Huu Phuoc Le, Régis Makhmara, and Ta Duy Nguyen. Fast genetic algorithms. In *Genetic and Evolutionary Computation Conference, GECCO 2017*, pages 777–784. ACM, 2017.

[DLN19] Duc-Cuong Dang, Per Kristian Lehre, and Phan Trung Hai Nguyen. Level-based analysis of the univariate marginal distribution algorithm. *Algorithmica*, 81:668–702, 2019.

[DN20] Benjamin Doerr and Frank Neumann, editors. *Theory of Evolutionary Computation—Recent Developments in Discrete Optimization*. Springer, 2020. Also available at https://cs.adelaide.edu.au/~frank/papers/TheoryBook2019-selfarchived.pdf.

[DNS17] Benjamin Doerr, Frank Neumann, and Andrew M. Sutton. Time complexity analysis of evolutionary algorithms on random satisfiable k-CNF formulas. *Algorithmica*, 78:561–586, 2017.

[DNSW11] Benjamin Doerr, Frank Neumann, Dirk Sudholt, and Carsten Witt. Runtime analysis of the 1-ANT ant colony optimizer. *Theoretical Computer Science*, 412:1629–1644, 2011.

[Doe19a] Benjamin Doerr. Analyzing randomized search heuristics via stochastic domination. *Theoretical Computer Science*, 773:115–137, 2019.

[Doe19b] Benjamin Doerr. 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 2019*, pages 1488–1496. ACM, 2019.

[Doe20a] Benjamin Doerr. Does comma selection help to cope with local optima? In *Genetic and Evolutionary Computation Conference, GECCO 2020*, pages 1304–1313. ACM, 2020.

[Doe20b] Benjamin Doerr. A gentle introduction to theory (for non-theoreticians). In *Genetic and Evolutionary Computation Conference, GECCO 2020, Companion Material*, pages 373–403. ACM, 2020.

[Doe20c] Benjamin Doerr. Probabilistic tools for the analysis of randomized optimization heuristics. In Benjamin Doerr and Frank Neumann, editors, *Theory of Evolutionary Computation: Recent Developments in Discrete Optimization*, pages 1–87. Springer, 2020. Also available at https://arxiv.org/abs/1801.06733.

[Doe20d] Carola Doerr. Complexity theory for discrete black-box optimization heuristics. In Benjamin Doerr and Frank Neumann, editors, *Theory of Evolutionary Computation: Recent Developments in Discrete Optimization*, pages 133–211. Springer, 2020. Also available at https://arxiv.org/abs/1801.02037.

[Doe21] Benjamin Doerr. Lower bounds for non-elitist evolutionary algorithms via negative multiplicative drift. *Evolutionary Computation*, 29:305–329, 2021.

[DT12] Kathryn A. Dowsland and Jonathan M. Thompson. Simulated annealing. In Grzegorz Rozenberg, Thomas Bäck, and Joost N. Kok, editors, *Handbook of Natural Computing*, pages 1623–1655. Springer, Berlin, Heidelberg, 2012.

[DWY21] Benjamin Doerr, Carsten Witt, and Jing Yang. Runtime analysis for self-adaptive mutation rates. *Algorithmica*, 83:1012–1053, 2021.

[DZ20] Benjamin Doerr and Weijie Zheng. Sharp bounds for genetic drift in estimation-of-distribution algorithms. *IEEE Transactions on Evolutionary Computation*, 24:1140–1149, 2020.

[ER63] Paul Erdős and Alfréd Rényi. On two problems of information theory. *Magyar Tudományos Akadémia Matematikai Kutató Intézet Közleményei*, 8:229–243, 1963.

[FKK16] Tobias Friedrich, Timo Kötzing, and Martin S. Krejca. EDAs cannot be balanced and stable. In *Genetic and Evolutionary Computation Conference, GECCO 2016*, pages 1139–1146. ACM, 2016.

[FS21] Mario Alejandro Hevia Fajardo and Dirk Sudholt. Self-adjusting offspring population sizes outperform fixed parameters on the cliff function. In *Foundations of Genetic Algorithms, FOGA 2021*, pages 5:1–5:15. ACM, 2021.

[GP14] Brian W. Goldman and William F. Punch. Parameter-less population pyramid. In *Genetic and Evolutionary Computation Conference, GECCO 2014*, pages 785–792. ACM, 2014.

[GW03] Oliver Giel and Ingo Wegener. Evolutionary algorithms and the maximum matching problem. In *Symposium on Theoretical Aspects of Computer Science, STACS 2003*, pages 415–426. Springer, 2003.

[HJKN08] Edda Happ, Daniel Johannsen, Christian Klein, and Frank Neumann. Rigorous analyses of fitness-proportional selection for optimizing linear functions. In *Genetic and Evolutionary Computation Conference, GECCO 2008*, pages 953–960. ACM, 2008.

[Hoe63] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58:13–30, 1963.

[HS18] Václav Hasenöhrl and Andrew M. Sutton. On the runtime dynamics of the compact genetic algorithm on jump functions. In *Genetic and Evolutionary Computation Conference, GECCO 2018*, pages 967–974. ACM, 2018.

[HY01] Jun He and Xin Yao. Drift analysis and average time complexity of evolutionary algorithms. *Artificial Intelligence*, 127:51–81, 2001.

[HY04] Jun He and Xin Yao. A study of drift analysis for estimating computation time of evolutionary algorithms. *Natural Computing*, 3:21–35, 2004.

[Jan11] Thomas Jansen. Simulated annealing. In Anne Auger and Benjamin Doerr, editors, *Theory of Randomized Search Heuristics*, pages 171–195. World Scientific Publishing, 2011.

[Jan13] Thomas Jansen. *Analyzing Evolutionary Algorithms – The Computer Science Perspective*. Springer, 2013.

[JJW05] Thomas Jansen, Kenneth A. De Jong, and Ingo Wegener. On the choice of the offspring population size in evolutionary algorithms. *Evolutionary Computation*, 13:413–440, 2005.

[JS98] Mark Jerrum and Gregory B. Sorkin. The metropolis algorithm for graph bisection. *Discrete Applied Mathematics*, 82:155–175, 1998.

[JS07] Jens Jägersküpper and Tobias Storch. When the plus strategy outperforms the comma strategy and when not. In *Foundations of Computational Intelligence, FOCI 2007*, pages 25–32. IEEE, 2007.

[JW07] Thomas Jansen and Ingo Wegener. A comparison of simulated annealing with a simple evolutionary algorithm on pseudo-Boolean functions of unitation. *Theoretical Computer Science*, 386:73–93, 2007.

[KWW09] Lars Kaden, Nicole Weicker, and Karsten Weicker. Metropolis and symmetric functions: a swan song. In *Evolutionary Computation in Combinatorial Optimization, EvoCOP 2009*, pages 204–215. Springer, 2009.

[Leh10] Per Kristian Lehre. Negative drift in populations. In *Parallel Problem Solving from Nature, PPSN 2010*, pages 244–253. Springer, 2010.

[Leh11] Per Kristian Lehre. Fitness-levels for non-elitist populations. In *Genetic and Evolutionary Computation Conference, GECCO 2011*, pages 2075–2082. ACM, 2011.

[Len20] Johannes Lengler. Drift analysis. In Benjamin Doerr and Frank Neumann, editors, *Theory of Evolutionary Computation: Recent Developments in Discrete Optimization*, pages 89–131. Springer, 2020. Also available at https://arxiv.org/abs/1712.00964.

[LN18] Per Kristian Lehre and Phan Trung Hai Nguyen. Level-based analysis of the population-based incremental learning algorithm. In *Parallel Problem Solving From Nature, PPSN 2018*, pages 105–116. Springer, 2018.

[LN19] Per Kristian Lehre and Phan Trung Hai Nguyen. On the limitations of the univariate marginal distribution algorithm to deception and where bivariate EDAs might help. In *Foundations of Genetic Algorithms, FOGA 2019*, pages 154–168. ACM, 2019.

[LOW19] Andrei Lissovoi, Pietro S. Oliveto, and John Alasdair Warwicker. On the time complexity of algorithm selection hyper-heuristics for multimodal optimisation. In *Conference on Artificial Intelligence, AAAI 2019*, pages 2322–2329. AAAI Press, 2019.

[LS18] Johannes Lengler and Angelika Steger. Drift analysis and evolutionary algorithms revisited. *Combinatorics, Probability & Computing*, 27:643–666, 2018.

[LS20] Per Kristian Lehre and Dirk Sudholt. Parallel black-box complexity with tail bounds. *IEEE Transactions on Evolutionary Computation*, 24:1010–1024, 2020.

[LW12] Per Kristian Lehre and Carsten Witt. Black-box search by unbiased variation. *Algorithmica*, 64:623–642, 2012.

[MB17] Vladimir Mironovich and Maxim Buzdalov. Evaluation of heavy-tailed mutation operator on maximum flow test generation problem. In *Genetic and Evolutionary Computation Conference, GECCO 2017, Companion Material*, pages 1423–1426. ACM, 2017.

[Mee07] Klaus Meer. Simulated annealing versus Metropolis for a TSP instance. *Information Processing Letters*, 104:216–219, 2007.

[MHS01] Matthias Müller-Hannemann and Stefan Schirra. *Algorithm Engineering*. Springer, 2001.

[MRR+53] Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21:1087–1092, 1953.

[MS17] Alberto Moraglio and Dirk Sudholt. Principled design and runtime analysis of abstract convex evolutionary search. *Evolutionary Computation*, 25:205–236, 2017.

[Müh92] Heinz Mühlenbein. How genetic algorithms really work: mutation and hillclimbing. In *Parallel Problem Solving from Nature, PPSN 1992*, pages 15–26. Elsevier, 1992.

[NOW09] Frank Neumann, Pietro S. Oliveto, and Carsten Witt. Theoretical analysis of fitness-proportional selection: landscapes and efficiency. In *Genetic and Evolutionary Computation Conference, GECCO 2009*, pages 835–842. ACM, 2009.

[NSW09] Frank Neumann, Dirk Sudholt, and Carsten Witt. Analysis of different MMAS ACO algorithms on unimodal functions and plateaus. *Swarm Intelligence*, 3:35–68, 2009.

[NW10] Frank Neumann and Carsten Witt. *Bioinspired Computation in Combinatorial Optimization – Algorithms and Their Computational Complexity*. Springer, 2010.

[OPH+18] Pietro S. Oliveto, Tiago Paixão, Jorge Pérez Heredia, Dirk Sudholt, and Barbora Trubenová. How to escape local optima in black box optimisation: when non-elitism outperforms elitism. *Algorithmica*, 80:1604–1633, 2018.

[OS14] Pietro S. Oliveto and Dirk Sudholt. On the runtime analysis of stochastic ageing mechanisms. In *Conference on Genetic and Evolutionary Computation, GECCO 2014*, pages 113–120, 2014.

[PHST17] Tiago Paixão, Jorge Pérez Heredia, Dirk Sudholt, and Barbora Trubenová. Towards a runtime comparison of natural and artificial evolution. *Algorithmica*, 78:681–713, 2017.

[RS14] Jonathan E. Rowe and Dirk Sudholt. The choice of the offspring population size in the $(1, \lambda)$ evolutionary algorithm. *Theoretical Computer Science*, 545:20–38, 2014.

[Rud97] Günter Rudolph. *Convergence Properties of Evolutionary Algorithms*. Verlag Dr. Kovăc, 1997.

[RW20] Amirhossein Rajabi and Carsten Witt. Self-adjusting evolutionary algorithms for multimodal optimization. In *Genetic and Evolutionary Computation Conference, GECCO 2020*, pages 1314–1322. ACM, 2020.

[SH88] Galen H. Sasaki and Bruce Hajek. The time complexity of maximum matching by simulated annealing. *Journal of the ACM*, 35:387–403, 1988.

[Sör15] Kenneth Sörensen. Metaheuristics – the metaphor exposed. *International Transactions on Operations Research*, 22:3–18, 2015.

[Sud13] Dirk Sudholt. A new method for lower bounds on the running time of evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 17:418–435, 2013.

[SW19] Dirk Sudholt and Carsten Witt. On the choice of the update strength in estimation-of-distribution algorithms and ant colony optimization. *Algorithmica*, 81:1450–1489, 2019.

[TGM06] Olivier Teytaud, Sylvain Gelly, and Jérémie Mary. On the ultimate convergence rates for isotropic algorithms and the best choices among various forms of isotropy. In *Parallel Problem Solving from Nature, PPSN 2006*, pages 32–41. Springer, 2006.

[Weg05] Ingo Wegener. Simulated annealing beats Metropolis in combinatorial optimization. In *Automata, Languages and Programming, ICALP 2005*, pages 589–601. Springer, 2005.

[Wit06] Carsten Witt. Runtime analysis of the (μ + 1) EA on simple pseudo-Boolean functions. *Evolutionary Computation*, 14:65–86, 2006.

[WZD21] Shouda Wang, Weijie Zheng, and Benjamin Doerr. Choosing the right algorithm with hints from complexity theory. In *International Joint Conference on Artificial Intelligence, IJCAI 2021*, pages 1697–1703. ijcai.org, 2021.

[ZCY21] Weijie Zheng, Huanhuan Chen, and Xin Yao. Analysis of evolutionary algorithms on fitness function with time-linkage property. *IEEE Transactions on Evolutionary Computation*, 25:696–709, 2021.

[ZZCY21] Weijie Zheng, Qiaozhi Zhang, Huanhuan Chen, and Xin Yao. When non-elitism meets time-linkage problems. In *Genetic and Evolutionary Computation Conference, GECCO 2021*, pages 741–749. ACM, 2021.