Choosing the Right Algorithm With Hints From Complexity Theory

Shouda Wang[1] , **Weijie Zheng**[2*] , **Benjamin Doerr**[1*]
[1] Laboratoire d'Informatique (LIX), École Polytechnique, CNRS, Institut Polytechnique de Paris, France
[2] Guangdong Provincial Key Laboratory of Brain-inspired Intelligent Computation, Department of Computer Science and Engineering, Southern University of Science and Technology, China
shouda.wang@polytechnique.edu, zhengwj13@tsinghua.org.cn, doerr@lix.polytechnique.fr

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 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 hill-climbers, evolutionary algorithms, or estimation-of-distribution algorithms (EDAs) have been very successful in solving optimization problems for which no established problem-specific algorithm exists. However, when faced with a novel optimization problem, a suitable choice among such large number of established heuristics is difficult. Since implementing and then adjusting a heuristic to the problem 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 heuristics has helped improve the understanding of these algorithms (see the textbook [Doerr and Neumann, 2020] or the tutorial [Doerr, 2020b]), has given suggestions for their parameter settings, and has even proposed new operators and algorithms, but we are not aware of direct attempts to aid the initial choice of the basic algorithm to be used.

What we propose in this work is a heuristic approach building on the notion of black-box complexity [Droste *et al.*, 2006]. 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 successfully used to obtain universal lower bounds. Knowing that the black-box complexity of the Needle problem is exponential [Droste *et al.*, 2006], we immediately obtain that no genetic algorithm or EDA can solve the Needle 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 $n \log n$ [Lehre and Witt, 2012] implies that many standard mutation-based evolutionary algorithms cannot optimize it faster.

With a positive perspective, black-box complexity has been used to 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 $O(n)$ [Doerr *et al.*, 2011], a novel crossover-based evolutionary algorithm was developed in [Doerr *et al.*, 2015]. Observing that the λ-parallel black-box complexity of the OneMax problem is only $O(\frac{n\lambda}{\log \lambda} + n \log n)$, a dynamic parameter choice giving superior runtimes was developed in [Badkobeh *et al.*, 2014].

In this work, we also aim at profiting from the guidance of black-box results, however not to design new algorithms, but to obtain an indication which of the existing algorithms could be useful for a particular problem. 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 algo-

*Corresponding Authors

rithms 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, 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 recently proposed DeceivingLeadingBlocks (DLB) problem [Lehre and Nguyen, 2019], for which only $O(n^3)$ runtime guarantees for several classic evolutionary algorithms [Lehre and Nguyen, 2019] and an $O(n^2 \log n)$ guarantee for the EDA *univariate marginal distribution algorithm (UMDA)* [Doerr and Krejca, 2020b] are known.

Finding efficient search heuristics for the DLB *problem:* The classic algorithms regarded in [Lehre and Nguyen, 2019] are all elitist or non-elitist but with parameter settings that let them imitate an elitist behavior. To obtain a first indication whether it is worth investigating non-elitist algorithms, we prove in Sect. 3 (i) that the $(1+1)$ elitist unbiased black-box complexity of the DLB problems is $\Omega(n^3)$ and (ii) that a simple, artificial, $(1+1)$-type non-elitist unbiased black-box algorithm can solve it in $O(n^2)$ time. These two findings motivate us to analyze the existing $(1+1)$-type non-elitist heuristics. Among them, we find that the Metropolis algorithm [Metropolis *et al.*, 1953] 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 Sect. 3.4), 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 show in Sect. 3.5 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 generate offspring using the information from at least two parents. That such algorithms can be superior, at least in principle, follows from our result in Sect. 4.1, which is an artificial crossover-based algorithm that solves DLB in time $O(n \log n)$. The working principles of this algorithm also include a learning aspect. 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 to sample good solutions, based on the information of many previously generated solutions. Hence, we focus on EDAs. We do not find a classic EDA for which we can prove a subquadratic runtime, but we succeed for the significance-based EDA [Doerr and Krejca, 2020a] and show in Sect. 4.2 that it optimizes DLB in time $O(n \log n)$ with high probability.

Overall, these results demonstrate that our heuristic theory-guided approach towards selecting good algorithms for a novel problem can indeed be fruitful. In particular, it suggests the Metropolis algorithm, for which very little rigorous support for preferring it over other algorithms existed previously. Our experimental analysis in Sect. 5 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. Sect. 6 concludes this paper.

For reasons of space, many details and all mathematical proofs had to be omitted. They can be found in a preprint soon to be submitted to the arXiv preprint server.

2 Preliminaries

In this paper, we consider pseudo-Boolean optimization problems, i.e. the maximization of functions $f : \{0,1\}^n \to \mathbb{R}$ where n is a positive integer. Inspired by evolutionary computation, we call $f(x)$ the fitness of the search point x.

2.1 Black-Box Optimization and Runtime

In (discrete) black-box optimization, we assume that the optimization algorithms can access only the fitness evaluation of search points for the problem to be solved. Classic black-box optimization algorithms include hill-climbers, the Metropolis algorithm, simulated annealing, evolutionary algorithms, and other bio-inspired search heuristics.

Unless a specific understanding of the problem at hand suggests to do otherwise, it is natural to look for algorithms that are invariant under the symmetries of the search space, as most algorithms have this property. The first to explicitly discuss such invariance properties for the search space $\{0,1\}^n$ of bit strings was the seminal paper [Lehre and Witt, 2012]. 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, but can only on the fitnesses of the search points generated. This also implies that all search points the algorithm has access to can only be generated from previous ones via unbiased variation operators. This observation 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 (also called *unary*) and crossover-based algorithms have an arity of two. We note that sampling a random search point is an unbiased operator with arity zero.

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*

- $\forall x^1, \dots, x^k, y, z \in \{0,1\}^n, \Pr[y = V(x^1, \dots, x^k)] = \Pr[y \oplus z = V(x^1 \oplus z, \dots, x^k \oplus z)]$,
- $\forall x^1, \dots, x^k, y \in \{0,1\}^n, \forall \sigma \in \mathcal{S}_n, \Pr[y = V(x^1, \dots, x^k)] = \Pr[\sigma(y) = V(\sigma(x^1), \dots, \sigma(x^k))]$, *where $\mathcal{S}_n$ represents the symmetric group on n letters.*

Algorithm 1 shows the k-ary unbiased black-box algorithm, and this work only considers unbiased algorithms.

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. The *runtime* $T := T_A(f)$ 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

Algorithm 1: Template of a k-ary unbiased black-box algorithm for optimizing f. Without explicit mention, we assume that each search point $x^{(t)}$ is evaluated immediately after being generated.

```
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));
```

is generated (and evaluated). In the notation of Algorithm 1, we have $T_A(f) = 1 + \inf\{t \mid x^{(t)} \in \arg\max f\}$.

2.2 Black-Box Complexity

To understand the difficulty of a problem for black-box optimization algorithms, inspired by classical complexity theory, [Droste *et al.*, 2006] defined the *black-box complexity* as the smallest (expected) number of function evaluations needed to solve a problem. More precisely, for a problem $\mathcal{F}$, that is, a set of functions $f : \{0,1\}^n \to \mathbb{R}$, the (unrestricted) black-box complexity is defined as $\mathrm{bbc}(\mathcal{F}) := \inf_A \sup_{f\in\mathcal{F}} E[T_A(f)]$, where A runs over all black-box algorithms in the infimum.

More meaningful complexities can be obtained by admitting only certain types of black-box algorithms. For example, by letting A run over all unary unbiased black-box algorithms, one obtains the unary unbiased black-box complexity, which answers the question of what is the best unary unbiased way to solve a given problem. In this work, we shall compare different restricted black-box notions to obtain a first indication of what types of established search heuristics might be efficient for a given problem.

2.3 The DLB Function and Known Runtimes

We now define the DLB function, which is the main object of our study and was first introduced in [Lehre and Nguyen, 2019] recently. 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 the DLB function is defined only for n even, let n in the remainder be an even integer.

For an $x \in \{0,1\}^n$, we call a pair of entries $(x_{2\ell+1}, x_{2\ell+2})$, $\ell \in [0..\frac{n-2}{2}]$, a block. If $x \neq 1^n$ then let (x_{2m+1}, x_{2m+2}) be the first block that is not 11, that is, $m = \inf\{\ell \mid x_{2\ell+1} \neq 1 \text{ or } x_{2\ell+2} \neq 1\}$. We call this block the *critical block* of x. If the critical block of x is 00, then $\mathrm{DLB}(x) := 2m+1$, otherwise (that is, if the critical block is 01 or 10), $\mathrm{DLB}(x) := 2m$. For $x = 1^n$, we define $\mathrm{DLB}(x) = n$. Hence DLB counts 2 for each 11 block on the left of the critical block and adds 1 if the critical block is 00. This makes all search points with critical block equal to 00 a local optimum.

We now review the most relevant known runtime results for this work. [Lehre and Nguyen, 2019] analyzed the classic mutation-based evolutionary algorithms $(1+\lambda)$ EA, $(\mu+1)$ EA, and (μ,λ) EA and proved $O(n^3)$ for each with the optimal parameter settings. They also proved that genetic algorithms using k-tournament selection, (μ,λ) selection, linear selection, or exponential ranking selection, also take an $O(n\lambda\log\lambda + n^3)$ expected runtime on the DLB problem.

From looking at the proofs in [Lehre and Nguyen, 2019], 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, 2020b] showed a $\Theta(n^3)$ runtime for $(1+1)$ EA. In Theorem 2, we extend this result to all $(1+1)$-elitist unary unbiased black-box algorithms.

As opposed to these polynomial runtime results, [Lehre and Nguyen, 2019] pointed out a potential weakness of the UMDA. They proved that the UMDA selecting μ fittest individuals from λ sampled individuals has an expected runtime of $e^{\Omega(\mu)}$ 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, 2020b] pointed out that the negative finding is caused by the unfortunate parameter choice and that with a population size large enough to prevent genetic drift [Sudholt and Witt, 2019] the UMDA solves the DLB efficiently: with 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.

3 From Elitist to Non-Elitist Algorithms

It is natural to start the search for a good heuristic with simple algorithms. As reviewed in Sect. 2.3, the $(1+1)$ EA with mutation rate $1/n$ can solve the DLB problem in expected time $O(n^3)$ [Lehre and Nguyen, 2019], but not faster [Doerr and Krejca, 2020b]. To see whether other $(1+1)$-elitist algorithms can do better, we first determine the $(1+1)$ elitist unbiased black-box complexity of the DLB problem. Noting that this is still $\Omega(n^3)$, we turn to non-elitist algorithms. We find an artificial non-elitist $(1+1)$-type algorithm and use it as inspiration to look for suitable established non-elitist heuristics. Unexpectedly, in the light of the previous literature (discussed in Sect. 3.4), we find that the Metropolis algorithm with constant temperature solves DLB in time $O(n^2)$.

3.1 Elitist $(1+1)$ Unbiased Black-Box Complexity

The elitist $(1+1)$ unbiased black-box complexity notion captures all algorithms which start with a random search point and then repeat (i) generating an offspring from the current search point via an unbiased (mutation) operator (possibly a different one in each iteration) and (ii) keeping as new search point the better of parent and offspring (with some tie-breaking in case of equal fitness) [Doerr and Lengler, 2017]. Unfortunately, we observe that this black-box complexity is $\Omega(n^3)$, which shows that to break the $O(n^3)$ barrier we have to work with larger population sizes or allow non-elitism.

Theorem 2. *The $(1+1)$-elitist black-box complexity of the* DLB *problem is* $\Omega(n^3)$.

The main argument of the proof of this result is a potential argument. We define the potential of a search point to be the number of leading 11-blocks plus $1-\frac{4}{n}$ for the critical block if it contains exactly one 1. For this potential, we show that any iteration of a $(1+1)$ elitist unbiased algorithm started with a search point of potential at least $\frac{n}{3}$ can increase the potential by at most $O(\frac{1}{n^2})$. Since a potential of $\frac{n}{2}$ is necessary to have the optimum, this gives the desired $\Omega(n^3)$ lower bound.

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

In Sect. 3.1, we conclude that to obtain a performance better than cubic, we need to ignore one of the restrictions: elitism, $(1+1)$-type, and unbiasedness. Omitting elitism appears the most natural since the previous lower bound proof heavily exploited the elitism of the algorithms regarded. To obtain a first indication of whether the class of $(1+1)$-type unbiased black-box algorithms contains interesting search heuristics for our DLB problem, we now determine an upper bound on the corresponding black-box complexity.

We easily observe that from a search point with critical block equal to 01 or 10, visible from an even DLB value, it suffices to flip a single bit to improve the fitness and at the same time reduce the Hamming distance to the optimum (and any one-bit flip that improves the fitness by at least two does reduce the Hamming distance). If the critical block is 00, then a one-bit flip reduces the Hamming distance if and only if the fitness worsens by only one (and there is no way to increase the fitness by flipping one bit).

These observations immediately suggest a simple unary unbiased black-box algorithm: Start with a random search point $x \in \{0,1\}^n$ and repeat (i) generating a new solution y by flipping a randomly chosen bit in x and (ii) accepting it (that is, setting $x := y$) if, with the above considerations, the fitness indicates to us that it is closer to the optimum than x. Repeat this until we have found the optimum.

Since each iteration has a chance of at least $\frac{1}{n}$ of reducing the distance to the optimum and the initial distance is at most n, the expected runtime of this artificial algorithm is $O(n^2)$.

Theorem 3. *The* $(1+1)$*-type unbiased black-box complexity of the* DLB *problem is* $O(n^2)$.

3.3 The Metropolis Algorithm Performs Well

Sect. 3.2 showed that there are, in principle, $(1+1)$-type unbiased algorithms which can optimize the DLB problem much faster than the cubic time which is best possible for elitist algorithms. The algorithm discussed in Sect. 3.2, of course, was highly artificial and overfitted to the DLB problem, but it suggests that there might also be established search heuristics solving the DLB problem faster than in cubic time. Given that the good performance above was made possible by the fact that the artificial algorithm was able to accept inferior solutions, the first natural choice for such a heuristic is the Metropolis algorithm. This simple $(1+1)$-type hill-climber can also accept inferior solutions, however only with a small probability that depends on the degree of inferiority and an algorithm parameter $\alpha \in (1,\infty)$. See Algorithm 2 for the precise pseudocode of the Metropolis algorithm.

Algorithm 2: Metropolis algorithm for maximizing f

```
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 x^(t) ← y with probability α^(f(y)-f(x^(t-1))) and
      x^(t) ← x^(t-1) otherwise;
```

The main result of this section is that the Metropolis algorithm can optimize DLB in quadratic time if the selection pressure is sufficiently high, that is, α is large enough.

Theorem 4. *The expected runtime of the Metropolis algorithm on the DLB problem is at most* $n^2/C(\alpha)$*, where* $C(\alpha) := \frac{2}{\alpha}\left(\frac{1}{2} - 2\sum_{k=1}^{\infty} k\alpha^{-2k}\right)$ *is a constant (depending only on* α*) which is positive when* $\alpha > \sqrt{2}+1$.

To prove this result, we need to argue that the negative effect of accepting inferior solutions, namely that solutions in higher distance from the optimum can be accepted, 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 define the potential of a search point as the number of leading 11 blocks, and this plus 0.25 when the critical block contains exactly one 1. We show that in each iteration (starting with a non-optimal search point) this potential in expectation increases by $\Omega(\frac{1}{n})$. Hence by additive drift theorem, it takes $O(n^2)$ time to reach a potential of $n/2$, which means that the optimum is found.

3.4 Literature Review on 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 it and non-elitist evolutionary algorithms. While it is generally believed that non-elitism can be helpful to leave local optima, there is surprisingly little evidence for this in terms of rigorous runtime analysis (at least in discrete search spaces).

The majority of the runtime analyses of the Metropolis algorithm on discrete problems does not suggest that this algorithm easily copes with local optima, at least not better than classic evolutionary algorithms. The Metropolis algorithm was proven to be able to compute approximate solutions for the maximum matching problem [Sasaki and Hajek, 1988] and to find the (unique) minimum bisection of random instance in the planted solution model [Jerrum and Sorkin, 1998]. [Wegener, 2005] provided a simple instance of the minimum spanning tree problem, which can be solved very efficiently by simulated annealing with a natural cooling schedule (or simple evolutionary algorithms), but for which the Metropolis algorithm with any temperature needs an exponential runtime. [Jansen and Wegener, 2007] analyzed the performance on the OneMax benchmark, but observed that the Metropolis algorithm is efficient only with very small temperatures (and also then does not beat simple hill-climbers or evolutionary algorithms). [Lissovoi *et al.*, 2019] showed that the Metropolis algorithm needs at least an expected number of $\tilde{\Omega}(n^{d-0.5})$ iterations to optimize the multimodal CLIFF benchmark with constant cliff length d (much worse than the $O(n^3)$ runtime of the move-acceptance hyper-heuristic) and at least $\exp(\Omega(n))$ time on the multimodal JUMP benchmark with jump size m (much slower than the $O(n^m)$ time of many evolutionary algorithms, e.g., [Droste *et al.*, 2002]). In the only result demonstrating that the Metropolis algorithm can have an advantage in coping with local optima, [Oliveto *et al.*, 2018] 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, and proved that the Metropolis algorithm can optimize this problem in time $n\alpha^{\Theta(d_1)} + \Theta(n\ell_2)$, whereas the $(1+1)$ EA needs time $\Omega(n^{\ell_1})$. What limits the generality of this result is that this valley is constructed onto a long path function, making this essentially a one-dimensional optimization problem.

In terms of the runtime analysis for non-elitist classic evolutionary algorithms, most of them either showed that if the selection pressure is high, then the non-elitist algorithm behaves very similar to its elitist counterpart [Lehre, 2011] or showed that if the selection pressure is low, then the algorithm cannot efficiently optimize any problem with unique optimum [Lehre, 2010]. For the (μ, λ) EA optimizing jump functions, the existence of a profitable middle regime was disproven in [Doerr, 2020a]. The strongest supports for non-elitism are [Jägersküpper and Storch, 2007], which showed that $(1, \lambda)$ EA with $\lambda \geq 5 \ln n$ optimizes CLIFF with length $n/3$ in time $\exp(5\lambda) \geq n^{25}$, [Dang *et al.*, 2021], which showed that the non-elitist EA with 3-tournament selection, $\lambda \geq c \log n$ for c a positive constant, and proper mutation parameter optimizes certain instances of the FUNNEL in expected runtime $O(n\lambda \log \lambda + n^2 \log n)$, and [Zheng *et al.*, 2021], which 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 $O(n^{3+c} \log n)$.

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

We now prove that no unary unbiased black-box algorithm can solve DLB faster than in quadratic time. This result is not strictly necessary for our heuristic approach of finding good established search heuristics, but adds a lot to the motivation to regard algorithms other than the ones with only mutation.

Theorem 5. *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. Our $\Omega(n^2)$ lower bound for all unary unbiased algorithms in Sect. 3.5 tells us that a better performance can only be found among algorithms that generate new solutions based on the information of more than one previous solution such as crossover-based genetic algorithms, binary differential evolution, or estimation-of-distribution algorithms (EDAs).

4.1 Higher-Arity Unbiased Black-Box Algorithms

To see how realistic it is to find a search heuristic solving DLB in time better than quadratic and to ideally also obtain an indication of how such algorithms could look like, we now ask what is the unbiased black-box complexity of DLB. From its structure, it is clear that a fast algorithm for this problem, once it has a solution x with the first m blocks set correctly (that is, with $\mathrm{DLB}(x) = 2m$), has to relatively quickly optimize the next block. Since no information can be gained about later blocks, an ideal algorithm focuses only on this block without wasting time on higher blocks (where nothing is to be gained at the moment) or lower blocks (where everything is done already). From this analysis, it is clear that such an algorithm has to *learn* which blocks are already set correctly (to avoid useless operations here) and it has to have a mechanism to quickly detect the next block. If the optimized blocks are learned correctly, the next relevant block can be identified by imitating a binary search. This can be done also in an unbiased fashion – all that is necessary is flipping half of the undetermined bits and seeing if this has a positive influence on the fitness. We spare the technical details and just note that binary operators are enough.

Theorem 6. *The binary unbiased black-box complexity of the* DLB *problem is* $O(n \log n)$.

4.2 Sig-cGA

Sect. 4.1 showed that higher-arity unbiased algorithms can optimize DLB more efficiently than unary ones. We have not found a classic crossover-based genetic algorithm with such an improved performance. The observation that learning what are the right bit values was an important aspect in Sect. 4.1 led us to focus on EDAs, the randomized search heuristics which try to evolve a probabilistic model of the search space in a way that finally good solutions are sampled with high probability. For classic EDAs, the learning (that is, the update of the probabilistic model) necessarily has to be relatively slow to prevent a genetic drift effect, see [Doerr and Zheng, 2020] and the references therein. To overcome this, the *significance-based compact genetic algorithm* (sig-cGA) has been proposed [Doerr and Krejca, 2020a], which uses a longer history to update the probabilistic model.

As other EDAs for the pseudo-Boolean problems, the sig-cGA uses *frequency vectors* $\tau \in [0,1]^n$ to describe the probabilistic model. A sample $x \in \{0,1\}^n$ from the corresponding model is a random search point such that $\Pr[x_i = 1] = \tau_i$ independently for all $i \in [1..n]$. Different from other EDAs, the sig-cGA only utilizes the frequencies $\frac{1}{n}$, $\frac{1}{2}$, and $1 - \frac{1}{n}$. Here the values $\frac{1}{n}$ and $1 - \frac{1}{n}$ indicate that the algorithm is relatively sure that the corresponding bit values should be sampled as 0 or 1, whereas the value $\frac{1}{2}$ indicates that such a decision cannot be made yet with sufficient certainty. The sig-cGA does not need other frequency values because it also uses, for each bit position $i \in [1..n]$, a history $H_i \in \{0,1\}^*$ of successful bit values. Only when this history gives a statistically significant indication that one of the bit values 0 or 1 leads to better solutions, is the corresponding frequency set to $\frac{1}{n}$ or $1 - \frac{1}{n}$.

More precisely, in each iteration two points are independently sampled from the current model. For each $i \in [1..n]$,

Algorithm 3: Sig-cGA for maximizing f

$t \leftarrow 0$; **for** $i \in [n]$ **do** $\tau_i^{(0)} \leftarrow \frac{1}{2}$ and $H_i \leftarrow \emptyset$;
repeat
  Sample x and y w.r.t. $\tau^{(t)}$. Let z be the winner w.r.t. f, chosen at random in case of a tie;
  for $i \in [n]$ **do**
    $H_i \leftarrow H_i \circ z_i$;
    if $\mathrm{sig}_\varepsilon(\tau_i^{(t)}, H_i) = \text{UP}$ **then** $\tau_i^{(t+1)} \leftarrow 1 - 1/n$;
    else if $\mathrm{sig}_\varepsilon(\tau_i^{(t)}, H_i) = \text{DOWN}$ **then** $\tau_i^{(t+1)} \leftarrow 1/n$;
    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*;

the bit values of the better one is stored in history H_i. If a statistical significance is detected in H_i, then i-th frequency is updated accordingly and H_i is emptied. We say that a significance of 1 (0 resp.) is detected when $\mathrm{sig}_\varepsilon(\tau_i^{(t)}, H_i) = \text{UP}$ (DOWN resp.), where function sig_ε is defined as follows. 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 $\|H[k]\|_1$ ($\|H[k]\|_0$ resp.) denote the number of ones (resp. zeros) 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 $\mathrm{sig}_\varepsilon(p, H) =$ (i) UP, if $p \in \{\frac{1}{n}, \frac{1}{2}\} \wedge \exists m \in \mathbb{N} : \|H[2^m]\|_1 \geq 2^m p + s(\varepsilon, 2^m p)$; (ii) DOWN, if $p \in \{\frac{1}{2}, 1 - \frac{1}{n}\} \wedge \exists m \in \mathbb{N} : \|H[2^m]\|_0 \geq 2^m(1-p) + s(\varepsilon, 2^m(1-p))$; (iii) STAY, else. See Algorithm 3 for the pseudocode.

The runtime of the sig-cGA was proven to be $O(n \log n)$ for both ONEMAX and LEADINGONES [Doerr and Krejca, 2020a], a performance not seen before with any unbiased search heuristic. Our *main result* in this section is that the sig-cGA also performs well on DLB and optimizes it in time $O(n \log n)$ with high probability as well. This is the first runtime analysis for the sig-cGA on a multimodal benchmark problem and the first indication that this algorithm can perform well also on such problems. The proof of this result (to be found in the extended version of this work) shows that in a typical run the frequencies of a block stay at $\frac{1}{2}$ until the ones of all block to the left have reached $1 - \frac{1}{n}$. From that moment on, these histories of the two bits in this block collect more ones than zeros, which leads to an update of the corresponding frequencies to $1 - \frac{1}{n}$ in logarithmic time. Then these frequencies stay at the high value with high probability.

Theorem 7. *The runtime of the sig-cGA with $\varepsilon > 6$ on* DLB *is $O(n \log n)$ with probability at least $1 - O\left(n^{2-\varepsilon/3} \log^2 n\right)$.*

5 Experiments

To compare the algorithms we ran the $(1+1)$ EA, UMDA, Metropolis algorithm, and sig-cGA on the DLB function for $n = 40, 80, \ldots, 200$. We used the standard mutation rate $p = 1/n$ for the $(1+1)$ EA, population sizes $\mu = 3n \ln n$ and $\lambda = 12\mu$ for the UMDA (as in [Doerr and Krejca, 2020b]), and temperature parameter $\alpha = 3$ (greater than $\sqrt{2} + 1$ as suggested in Theorem 4) 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 and 80.

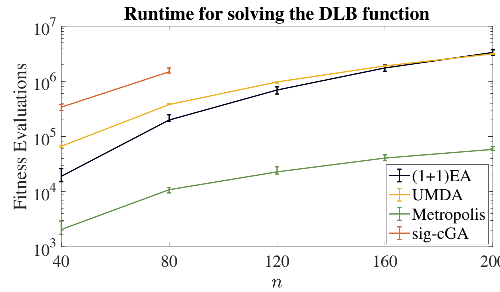

Figure 1: The median number of fitness evaluations (with the first and third quartiles) of $(1+1)$ EA, UMDA, Metropolis algorithm, and the sig-cGA on DLB with $n \in \{40, 80, 120, 160, 200\}$ in 20 independent runs (10 runs and $n \in \{40, 80\}$ for the sig-cGA).

Our experiments clearly show an excellent performance of the Metropolis algorithm, whereas the two EDAs perform much worse than what the asymptotic results suggest.

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 existed before. Both a mathematical runtime analysis and experiments proved it to be significantly superior to all previously analyzed algorithms for DLB.

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, 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

[Badkobeh *et al.*, 2014] Golnaz Badkobeh, Per Kristian Lehre, and Dirk Sudholt. Unbiased black-box complexity of parallel search. In *PPSN 2014*, pages 892–901. Springer, 2014.

[Dang *et al.*, 2021] Duc-Cuong Dang, Anton Eremeev, and Per Kristian Lehre. Escaping local optima with non-elitist evolutionary algorithms. In *AAAI 2021*. AAAI Press, 2021. To appear.

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

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

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

[Doerr and Neumann, 2020] 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.

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

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

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

[Doerr, 2020a] Benjamin Doerr. Does comma selection help to cope with local optima? In *GECCO 2020*, pages 1304–1313. ACM, 2020.

[Doerr, 2020b] Benjamin Doerr. A gentle introduction to theory (for non-theoreticians). In *GECCO 2020 Companion*, pages 373–403, 2020.

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

[Droste *et al.*, 2006] 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.

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

[Jansen and Wegener, 2007] 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.

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

[Lehre and Nguyen, 2019] 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 *FOGA 2019*, pages 154–168. ACM, 2019.

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

[Lehre, 2010] Per Kristian Lehre. Negative drift in populations. In *PPSN 2010*, pages 244–253. Springer, 2010.

[Lehre, 2011] Per Kristian Lehre. Fitness-levels for non-elitist populations. In *GECCO 2011*, pages 2075–2082. ACM, 2011.

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

[Metropolis *et al.*, 1953] 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.

[Oliveto *et al.*, 2018] 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.

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

[Sudholt and Witt, 2019] 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.

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

[Zheng *et al.*, 2021] Weijie Zheng, Qiaozhi Zhang, Huanhuan Chen, and Xin Yao. When non-elitism meets time-linkage problems. In *GECCO 2021*. ACM, 2021. https://doi.org/10.1145/3449639.3459347.