Elitist Black-Box Models: Analyzing the Impact of Elitist Selection on the Performance of Evolutionary Algorithms

Carola Doerr
CNRS & Univ. Pierre et Marie Curie
Paris, France

Johannes Lengler
ETH Zürich
Zürich, Switzerland

ABSTRACT

Black-box complexity theory provides lower bounds for the runtime of black-box optimizers like evolutionary algorithms and serves as an inspiration for the design of new genetic algorithms. Several black-box models covering different classes of algorithms exist, each highlighting a different aspect of the algorithms under considerations. In this work we add to the existing black-box notions a new *elitist black-box model*, in which algorithms are required to base all decisions solely on (a fixed number of) the best search points sampled so far. Our model combines features of the ranking-based and the memory-restricted black-box models with elitist selection.

We provide several examples for which the elitist black-box complexity is exponentially larger than that the respective complexities in all previous black-box models, thus showing that the elitist black-box complexity can be much closer to the runtime of typical evolutionary algorithms.

We also introduce the concept of p-Monte Carlo black-box complexity, which measures the time it takes to optimize a problem with failure probability at most p. Even for small p, the p-Monte Carlo black-box complexity of a function class $\mathcal{F}$ can be smaller by an exponential factor than its typically regarded Las Vegas complexity (which measures the *expected* time it takes to optimize $\mathcal{F}$).

Categories and Subject Descriptors

F.2.2 [**Theory of Computation**]: Analysis of Algorithms and Problem Complexity—*Nonnumerical Algorithms and Problems*

Keywords

Black-Box Complexity; Elitist Selection; Comparison-Based Algorithms; Evolutionary Computation

1. INTRODUCTION

Black-box models are classes of algorithms that are designed to help us understand how efficient commonly used search strategies like evolutionary algorithms (EAs) and other randomized search heuristics (RSHs) are. Several models exist, each designed to analyze a different aspect of search heuristics. For example, the memory-restricted model [10, 16] helps us to understand the influence of the population size on the efficiency of the search strategy, while the ranking-based black-box model [11, 17, 27] analyzes how much a heuristic loses by not using absolute but merely relative fitness values.

Having been introduced to the evolutionary computation community in [14, 16], black-box complexity is a young but highly active area of current research efforts [1, 3, 4, 6–8, 10–12, 17, 21, 22, 26, 27]. The insights from black-box complexity studies can be used to design more efficient genetic algorithms, as the recent $(1 + (\lambda, \lambda))$ GA from [5] shows.

We contribute to the existing literature a new model, which we call the *elitist black-box model*. As the name suggests, our model is designed to analyze the effect of elitist selection rules on the performance of search heuristics. In evolutionary computation, elitist selection is very common as it can be seen as a literate interpretation of the "survival of the fittest" principle in an optimization context. We remark already here that the usage of "elitist selection" is not standardized in the EA literature. Some subcommunities would therefore rather call our elitist black-box model a *black-box model with truncation selection*. Thus, to be precise, the algorithms in our $(\mu + \lambda)$ elitist model are required to select from the offspring and the parent population some μ search points of largest fitness values.

1.1 Previous Work

In contrast to classical complexity notions discussed in the computer science literature, black-box complexity focuses on so-called black-box optimizers, i.e., algorithms that do not have access to the function at hand other than by evaluating possible solutions, which are referred to as *search points*. If $\mathcal{C}$ is a class of black-box optimizers, then the $\mathcal{C}$-black-box complexity of a class $\mathcal{F}$ of functions is measured by the expected number of function evaluations that are needed by a best possible algorithm in $\mathcal{C}$ to optimize any instance $f \in \mathcal{F}$ (formal definitions will be given in Section 2). This number is a lower bound for the efficiency of any algorithm $\mathcal{A} \in \mathcal{C}$ and thus helps us understand how adequate an algorithmic choice is for the given problem class $\mathcal{F}$.

Among the most important algorithmic choices in the design of evolutionary algorithms are the population size, the sampling strategies (often called variation operators), and the selection rules. Existing black-box models cover these aspects in the following way. While the *memory-restricted*

GECCO '15, July 11 - 16, 2015, Madrid, Spain
© 2015 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ISBN 978-1-4503-3472-3/15/07...$15.00
DOI: http://dx.doi.org/10.1145/2739480.2754654

model [10,16] and the *parallel* black-box model [3,4] analyze the influence of the population-size, the *unbiased* model [8, 22,26] considers the efficiency of search strategies using only so-called unbiased variation operators. The influence of the selection rules have been analyzed in the *comparison-based* and *ranking-based* black-box model [11,17,27], with a focus on not revealing full fitness information to the algorithm but rather the comparison or the ranking of search points. The idea behind these models is that, in contrast to other search strategies like the physics-inspired simulated annealing, many evolutionary algorithms base their selection solely on *relative* and not on *absolute* fitness values.

While the comparison-based and the ranking-based models provide only relative fitness values, they do not require the algorithms to always select the better ones. Search strategies that adhere to this selection rule are called *elitist* algorithms in the evolutionary computation literature. Many common and widely applied black-box optimization strategies like $(\mu + \lambda)$ and (μ, λ) EAs as well as local hill climbers such as Randomized Local Search (RLS) are of this type. On the other hand, many practical algorithms intentionally keep suboptimal solutions to enhance population diversity, or to better explore the search space [28,29]. It has been shown that in some situations, specific elitist algorithms like RLS or the $(\mu + 1)$ EA are inferior to non-elitist algorithms [18,20,25]. In this paper, we go one step further and investigate the performance of *all* elitist algorithms simultaneously.

As mentioned before, algorithms in the ranking-based and comparison-based models do not need to be elitist. (Ab-)Using this, algorithms can be designed in both models that have much smaller runtimes than typical EAs [10,11]. As we shall see in Section 3 this even remains true if we regard (1+1) memory-restricted unary unbiased comparison-based algorithms, which constitutes the most restrictive combination of the restrictions from existing black-box models. We will see that such algorithms can crucially profit from eventually giving preference to search points of fitness inferior to that of the current best search points.

1.2 Our Model, New Complexity Measures, and Results

We provide in this work a model to analyze the impact of elitist selection on the runtime of black-box optimizers. In this *elitist black-box model* the population of the algorithms may contain only search points of best-so-far fitness values. That is, if the population size is μ, then at any point in time only the μ best-so-far search points (of possibly different fitness values) are allowed to be kept in the population (ties broken arbitrarily). All other previously sampled search points are not allowed to influence the behavior of the algorithm any more. We show (Section 3) that already for quite simple function classes there can be an exponential gap between the efficiency of elitist and non-elitist black-box algorithms. We also show (Section 4) that some shortcomings of previous models can be eliminated when they are combined with an elitist selection requirement. More precisely we show that the elitist unary unbiased black-box complexity of $\textsc{Jump}_k$ is of order $\Omega(n^k)$ and thus non-polynomial for $k = \omega(1)$. In contrast, the unary unbiased black-box complexity of $\textsc{Jump}_k$ is known to be polynomial even for extreme values of k [6].

In previous models, the black-box complexity has been defined in a *Las Vegas* manner, i.e, it measures the expected number of function evaluations until the algorithm hits the optimum. On the other hand, many proofs actually show that with high (or constant) probability the optimum is found after a certain number of steps, and then *random restarts* are used to bound the expected runtime. In (the strict version of) the elitist model the algorithms are not allowed to do random restarts, since new search points can be kept in the population only if they are among the μ best ones sampled so far. Since this is a rather artificial problem (many real-world optimization routines make use of restarts), we introduce in this work the concept of *Monte Carlo black-box complexities.* Roughly speaking, the p-Monte Carlo black-box runtime of a black-box algorithm A on a function f is the minimal number of queries A needs in order to find the optimum of f with probability at least $1 - p$. The complexity class is then derived in the usual way, cf. Section 2.1. We regard in our work both Monte Carlo complexities and standard (i.e., *Las Vegas*) complexities. For elitist black-box algorithms these two notions can differ substantially as we shall see in Section 3.1.

In the following we consider only discrete search spaces, and even more restrictively, only pseudo-Boolean functions $f : \{0,1\}^n \to \mathbb{R}$. Generalizations to non-finite or continuous search spaces are straightforward.

2. THE ELITIST BLACK-BOX MODEL

The *elitist black-box model* covers all algorithms that follow the pseudo-code in Algorithm 1. That is, a $(\mu + \lambda)$ *elitist black-box algorithm* is initialized by sampling μ search points. In each subsequent round it samples λ new search points from distributions that depend only on the current population X and the ranking of X with respect to the fitness function f.[1] When all λ search points have been evaluated, the algorithm proceeds to the selection step. A newly created search point replaces a point in X if its fitness is at least as good as one of the search points in X; it is discarded otherwise. In case of ties we allow the algorithm to select the search points it prefers.

The elitist black-box model covers many common EAs such as $(\mu + \lambda)$ EAs, Randomized Local Search (RLS), and other hill climbers. It does not cover algorithms with non-elitist selection rules like tournament or fitness-proportional selection.

Several extensions and variants of the model are possible, including in particular one in which the μ first search points cannot be sampled adaptively, where the selection has to be unbiased among search points of the same rank, where only offsprings can be selected (comma strategies), or a non-ranking-based one where absolute instead of relative fitness information is provided. Note that the latter would allow for fitness-dependent mutation rates, which are excluded by the variant analyzed here. The lower bounds presented in Sections 3 and 4 actually hold for this non-ranking-based model (and are thus even more powerful than such only applicable to the model described in Algorithm 1). The model can certainly also be extended to an unbiased elitist one, in which the distribution $p^{(t)}$ in line 7 of Algorithm 1 has to be

[1]Two search points have the same rank if and only if they have the same fitness. The search points of X with maximal f-values are rank one, the ones with second largest f-values are rank two etc.

Algorithm 1: The $(\mu + \lambda)$ elitist black-box algorithm for maximizing an unknown function $f : \{0,1\}^n \to \mathbb{R}$

```
1 Initialization:
2   $X \leftarrow \emptyset$;
3   for $i = 1, \ldots, \mu$ do
4       Depending only on the multiset $X$ and the ranking $\rho(X, f)$ of $X$ induced by $f$, choose a probability distribution $p^{(i)}$ over $\{0,1\}^n$ and sample $x^{(i)}$ according to $p^{(i)}$;
5       $X \leftarrow X \cup \{x^{(i)}\}$;
6 Optimization: for $t = 1, 2, 3, \ldots$ do
7     Depending only on the multiset $X$ and the ranking $\rho(X, f)$ of $X$ induced by $f$ choose a probability distribution $p^{(t)}$ on $(\{0,1\}^n)_{i=1}^{\lambda}$ and sample $(y^{(1)}, \ldots, y^{(\lambda)})$ according to $p^{(t)}$;
8     Set $X \leftarrow X \cup \{y^{(1)}, \ldots, y^{(\lambda)}\}$;
9     for $i = 1, \ldots, \lambda$ do  Select $x \in \arg\min X$ and update $X \leftarrow X \setminus \{x\}$;
```

unbiased in the sense of Lehre and Witt [22]. See Section 4 for results on the unbiased elitist model.

Note that elitist black-box algorithms covered by Algorithm 1 are *memory-restricted* in the sense of [10, 16], that is, they cannot store any other information than the current population and its ranking. All information about previous search points (e.g., their number) has to be discarded. The (1+1) version of the elitist model is *comparison-based* (that is, a query reveals only if an offspring has worse, equal, or better fitness than its parent), while the $(\mu+\lambda)$ versions are *ranking-based* in the sense of [11]. This means that the algorithm has no information about absolute fitness values, but only knows how the fitnesses of the $(\mu + \lambda)$ search points compare to each other. For memory-restricted algorithms these two notions coincide, while for non-memory-restricted models the ranking-based complexity can be smaller by a logarithmic factor than the comparison-based complexity, since in the ranking-based model the algorithm always gets a full ranking of the previously sampled search points while in the comparison-based model it has to specify two search points whose fitness is compared with each other.[2]

2.1 Monte Carlo vs. Las Vegas Black-Box Complexities

As discussed in Section 1.2, usually the black-box complexity of a function class $\mathcal{F}$ is defined in a *Las Vegas* manner (measuring the *expected number* of function evaluations), while in the case of elitist black-box complexity we also introduce a *p-Monte Carlo black-box complexity*, where we allow some failure probability p (see below for formal definitions). If we make a statement about the Monte Carlo complexity without specifying p, then we mean that for *every constant* $p > 0$ the statement holds for the p-Monte Carlo complexity. However, we sometimes also regard p-Monte Carlo complexities for non-constant $p = p(n) = o(1)$, thus yielding high probability statements.

For most black-box complexities, the Las Vegas and the Monte Carlo notions are closely related: every Las Vegas algorithm is also (up to a factor of $1/p$ in the runtime) a p-Monte Carlo algorithm by Markov's inequality, and a Monte Carlo algorithm can be turned into a Las Vegas algorithm by restarting the algorithm until the optimum is found. In particular, if restarts are allowed then Las Vegas and Monte Carlo complexities differ by at most a constant factor. This has been made explicit in [7, Remark 2] and is heavily used there as well as in a number of other results on black-box complexity. It is not difficult to see that such a reasoning fails for elitist black-box algorithms, as they are not allowed to do arbitrary restarts: if the sampled solution intended for a restart is not as good as the ones currently in the memory, it has to be discarded (line 9 of Algorithm 1). Las Vegas and Monte Carlo elitist black-box complexities may therefore differ significantly from each other, see Section 3.1 for an example with exponentially large gap.

We come to the formal definition. Let $\mathcal{F}$ be a class of pseudo-Boolean functions, and let $p \in [0, 1)$. The *Las Vegas complexity* of an algorithm A for $\mathcal{F}$ is the maximum expected number of function evaluations of f before A evaluates an optimal search point for the first time, where the maximum is taken over all $f \in \mathcal{F}$. The Las Vegas complexity of $\mathcal{F}$ with respect to a class $\mathcal{A}$ of algorithms is the minimum ("best") Las Vegas complexity among all $A \in \mathcal{A}$ for $\mathcal{F}$. The p-Monte Carlo complexity of $\mathcal{F}$ with respect to $\mathcal{A}$ is the minimum number T such that there is an algorithm in $\mathcal{A}$ which has for all $f \in \mathcal{F}$ a probability of at least $1-p$ to find an optimum with T function evaluations. The $(\mu + \lambda)$ *elitist Las Vegas (elitist p-Monte Carlo) black-box complexity* of $\mathcal{F}$ is the Las Vegas (p-Monte Carlo) complexity of $\mathcal{F}$ with respect to the class of all $(\mu + \lambda)$ elitist black-box algorithms.

To ease terminology, we will say that an algorithm *spends time t* on a function f if it uses at most t function evaluations on f. Moreover, we call the *runtime* of an algorithm A on a function f to be the expected number of function evaluations of A until it evaluates an optimal search point. In this way, the Las Vegas complexity of $\mathcal{A}$ on $\mathcal{F}$ is the worst-case (over all $f \in F$) runtime of the best algorithm $A \in \mathcal{A}$.

If we are interested in the asymptotic p-Monte Carlo complexity of an algorithm A on a function class $\mathcal{F}$ then we will frequently make use of the following observation, which follows from Markov's inequality and the law of total expectation. We omit the proof.

Remark 1. *Let $p \in (0, 1)$. Assume that there is an event $\mathcal{E}$ of probability $p_{\mathcal{E}} < p$ such that conditioned on $\neg\mathcal{E}$ the algorithm A finds the optimum after* expected *time at most T. Then the p-Monte Carlo complexity of A on f is at most $(1 - p_{\mathcal{E}})(p - p_{\mathcal{E}})^{-1}T$. In particular, if $p - p_{\mathcal{E}} = \Omega(1)$ then the p-Monte Carlo complexity is $O(T)$.*

2.2 (Non-)Applicability of Yao's Principle

A convenient tool in black-box complexity theory is Yao's Principle. In simple words, Yao's Principle allows to restrict one's attention to the runtime T of a best-possible *deterministic* algorithms on a *random* input instead of regarding the best-possible performance of a *random* algorithm on an *arbitrary* input. Analyzing the former is often considerably easier than directly bounding the performance of any possible randomized algorithm. Yao's Principle states that T is a

[2]Binary search shows that the gap between the two notions cannot be larger than a logarithmic factor, but on the other hand this logarithmic gap occurs already for the simple OneMax problem. While it's ranking-based black-box complexity is only $\Theta(n/\log n)$ [11], its comparison-based complexity is $\Theta(n)$ by a straightforward application of Yao's Principle.

lower bound for the expected performance of a best possible randomized algorithm for the regarded problem. Formally, Yao's Principle is the following.

LEMMA 2 (YAO'S PRINCIPLE [30]). *Let Π be a problem with a finite set $\mathcal{I}$ of input instances (of a fixed size) permitting a finite set $\mathcal{A}$ of deterministic algorithms. Let p be a probability distribution over $\mathcal{I}$ and q be a probability distribution over $\mathcal{A}$. Then,*

$$\min_{A\in\mathcal{A}} \mathrm{E}[T(I_p,A)] \le \max_{I\in\mathcal{I}} \mathrm{E}[T(I,A_q)]\,, \tag{1}$$

where I_p denotes a random input chosen from $\mathcal{I}$ according to p, A_q a random algorithm chosen from $\mathcal{C}$ according to q and $T(I,A)$ denotes the runtime of algorithm A on input I.

It is interesting to note that the informal interpretation of Yao's Principle given above does not apply to elitist algorithms. To illustrate this phenomenon, let us consider the (1+1) elitist model, though the argument can be easily extended to population-based elitist algorithms. Let p be the uniform distribution over the instances

$$\mathrm{OM}_z : \{0,1\}^n \to \mathbb{R}, x \mapsto n - \sum_{i=1}^{n} (x_i \oplus z_i), \tag{2}$$

$z \in \{0,1\}^n$, of the well-known ONEMAX problem (see Section 3 for some background on this problem). Let A be any deterministic algorithm. Then A has a positive (in fact, fairly large) probability during the optimization of I_p of getting stuck in some search point x: As a deterministic (1+1) elitist algorithm there exists a search point $y = y(x)$ such that whenever the algorithm sees x in the memory it samples y next. If the OM_z-fitness of y is strictly smaller than that of x, offspring y has to be discarded immediately, in which case the algorithm is in exactly the same situation as before. It can thus never escape from x, and the expected runtime of the algorithm on OM_z is infinite. This situation happens frequently and indeed it is not difficult to show that every deterministic (1+1) elitist algorithm has an infinite expected runtime on a uniformly chosen ONEMAX instance. The lower bound in (1) is thus infinite, too, suggesting that the elitist black-box complexity of this problem is infinite as well. However, there are simple elitist randomized search strategies that have finite expected runtime on ONEMAX, for example, RLS and the $(1+1)$ EA.

Why does this example not contradict Yao's Principle? Reading Lemma 2 carefully, we see that it makes a statement only about such randomized algorithms that are a convex combination of deterministic ones. In other words, the randomized algorithms (on a fixed input size) are given by making one random choice at the beginning, determining which of the finitely many deterministic algorithms we apply. For typical classes of algorithms *every* randomized algorithm is such a convex combination of deterministic algorithms (and randomized algorithms are, in fact, often defined this way). In this case Yao's Principle can be summarized in the way we described before Lemma 2, i.e., as a statement that links the worst-case expected runtime of randomized algorithms with the best expected runtime of deterministic algorithms on random input. The previous paragraph, however, explains that in the elitist black-box model there are randomized algorithms which cannot be expressed as a convex combination of deterministic ones. For this reason, we can never apply Yao's Principle directly to the class of elitist black-box algorithms. Similar considerations hold for other classes of memory-restricted black-box algorithms, but have not been mentioned explicitly in the literature. We are not aware of any other class of evolutionary algorithm where such an anomaly occurs and find the putative non-applicability of Yao's Principle quite noteworthy.

Due to the problems outlined above, we will often consider in our lower bound proofs a superset $\mathcal{A}'$ of algorithms which contains all elitist ones and which has the property that every randomized algorithm in $\mathcal{A}'$ can be expressed as a convex combination of deterministic ones. A lower bound shown for this broader class trivially applies to all elitist black-box algorithms. Observe in particular that in a class of black-box algorithms where every algorithm knows the number of previous steps, every randomized strategy is a convex combination of deterministic strategies: the algorithm can just flip all coins in advance, and essentially use the i-th random bit (or bit string) in the i-th step. Hence, such classes may be used to apply Yao's Principle. Note that still we obtain lower bounds that are much stronger than those of any of the previous black-box models.

3. EXPONENTIAL GAPS TO PREVIOUS MODELS

We provide some function classes for which the elitist black-box complexity is exponentially larger than their black-box complexities in any of the previously regarded models. The basis for these functions is ONEMAX, one of the best studied example functions in the theory of evolutionary computation. The original ONEMAX function simply counts the number of ones in a bitstring. Maximizing ONEMAX thus corresponds to finding the all-ones string.

Search heuristics are typically invariant with respect to problem encoding, and as such they have the same runtime for any function from the generalized ONEMAX function class ONEMAX $:= \{\mathrm{OM}_z \mid z \in \{0,1\}^n\}$, where OM_z is defined by (2). We call z, the unique global optimum of function OM_z, the *target string* of OM_z.

A very simple heuristic optimizing ONEMAX in $\Theta(n \log n)$ steps is *Randomized Local Search* (RLS). Since a variant of RLS will be used in both our upper bounds, we briefly recall that this algorithm is initialized with a uniform sample x. In each iteration one bit position $j \in [n]$ is chosen uniformly at random. The j-th bit of x is flipped and the fitness of the resulting search point y is evaluated. The better of the two search points x and y is kept for future iterations (favoring the newly created individual in case of ties). As is easily verified, RLS is a unary unbiased (1+1) elitist black-box algorithm.

3.1 The Double OneMax Problem

Our first example is a class of functions with two local optima that are far apart from each other, but both of which can be found efficiently by standard heuristics. As mentioned we will show that for such problems there can be an exponential gap between the elitist black-box complexity and the black-box complexity in a model in which all algorithms have to be unbiased, memory-restricted with size bound one, and purely comparison-based. This shows that our model strengthens the existing landscape of black-box models considerably. The example also shows that the Las

Vegas complexity of a problem can be exponentially larger than its Monte Carlo complexity.

The example is surprisingly simple and can easily be extended to similar functions with two local optima that are sufficiently far apart. To keep things simple we consider the following concrete class of functions. For a string $z^1 \in \{0,1\}^n$ and a set $S \subseteq [n]$ let $z^2 := z^2(z^1, S)$ be the string whose entries in $[n] \setminus S$ equal those of z^1 while the other entries are inverted. That is, $z^2 = z^1 \oplus e_S$, where here and in the following e_S is the binary vector whose entries in positions $i \in S$ are one and all other entries are zero. Let

$$\text{OM}_{z^1,S} := \begin{cases} \max\{\text{OM}_{z^1}(x), \text{OM}_{z^2}(x)\}, & \text{if } x \neq z^2, \\ n+1, & \text{otherwise.} \end{cases}$$

The unique global optimum of this function is z^2, while the search point z^1 is a local optimum whenever $|S| > 1$. For all $x \notin \{z^1, z^2\}$ the fitness $\text{OM}_{z^1,S}(x)$ equals $\text{OM}_{z^2,S}(x)$. Unless the algorithm queries either z^1 or z^2 it can therefore not distinguish between the two functions.

We consider the class of functions $\mathcal{F} := \{\text{OM}_{z^1,S} \mid z^1 \in \{0,1\}^n, S \subseteq [n] \text{ with } |S| = n/2\}$ and show the following.

Theorem 3. *Let $\varepsilon > 0$. The (1+1) elitist $(1/2+\varepsilon)$-Monte Carlo black-box complexity of $\mathcal{F}$ and its unary unbiased, (1+1)-memory restricted, comparison-based black-box complexity is $O(n \log n)$, while the (1+1) elitist Las Vegas black-box complexity of $\mathcal{F}$ is exponential in n even if the algorithm has access to the absolute fitness values.*

The key idea beyond Theorem 3 is the fact that in all previous models an algorithm that gets trapped by the wrong local optimum z^1 can simply do a restart and with constant probability it will climb up a path leading to z^2 in the next trial. Indeed, a simple version of Randomized Local Search with random restarts will achieve the $O(n \log n)$ upper bound. On the other hand, an elitist black-box algorithm needs to test either z^1 or z^2, and may be trapped there with (almost) no more information than that the true optimum is at distance $n/2$. Since it cannot do a simple restart, it (roughly speaking) has to sample points at distance $n/2$ until it eventually finds the optimum.

The example in Theorem 3 also shows that there can be an exponential gap between the Monte Carlo and the Las Vegas complexity of a class $\mathcal{F}$ of functions. Indeed, it is easy to see that the probability of RLS to have found the optimum of the functions within some $O(n \ln n)$ iterations is $(1 - o(1))1/2$.

Proof of Theorem 3. For the upper bound, consider the following random local search algorithm (RLS) with random restarts. We start with a uniform sample x. In each iteration, with probability $1/(10n \log n)$ the algorithm restarts, i.e., replaces x by a a new uniform sample. Otherwise, one bit position $j \in \{1, 2, \dots, n\}$ is chosen uniformly at random. The j-th bit of x is flipped and the fitness of the resulting search point y is evaluated. The better of the two search points x and y replaces x for future iterations (favoring y in case of ties). We show that this algorithm has expected optimization time $O(n \log n)$.

Whenever $x \notin \{z^1, z^2\}$ then the one-bit flip has probability at least $n - f(x)$ to increase the fitness of x (this can be proven by an easy case distinction whether or not $\text{OM}_{z^1}(x) \geq \text{OM}_{z^2}(x)$). This is at least as large as the progress probability for OneMax. Therefore, if no restart happens in, say, $5n \log n$ steps (which is true with constant probability) then with high probability RLS finds a local optimum of $\text{OM}_{z^1,S}$ in this time. In this case, the local optimum will be the global one with probability $1/2$. Summarizing, after each restart, the algorithm has at least a constant probability to find the global optimum in the next $5n \log n$ steps. This proves the first claim in Theorem 3.

For the lower bound, we essentially show that every algorithm (that terminates) ends up in z^1 with probability $1/2$, at which point an elitist algorithm only knows that the true optimum z^2 is (roughly) randomly distributed among the search points at distance $n/2$ of z^1. Since it can never "escape" from z^1 (unless it eventually samples z^2), the expected time after having reached z^1 is exponential in n.

To turn this intuition into a formal proof, we employ Yao's Principle (Lemma 2). As described in Section 2.2, we need to consider a larger class $\mathcal{A}$ of algorithms defined as follows. We call the time until the algorithm queries for the first time z^1 or z^2 "first phase", and the remaining time "second phase". Since we are not too much interested in the time that an algorithm spends in the first phase, we simply give away to the algorithm the two local optima z^1 and z^2. That is, during the first phase the algorithm may access the set $\{z^1, z^2\}$. We also allow the algorithm to store all search points that it samples throughout this phase. During this phase every randomized algorithm is a convex combination of deterministic ones, so we may use Yao's Principle and restrict ourselves to an algorithm A that is deterministic in the first phase. We assume that the problem instance $\text{OM}_{z^1,S}$ is taken from $\mathcal{F}$ uniformly at random.

Note that in the first phase the algorithm does not gain any additional information by querying any search point $x \notin \{z^1, z^2\}$ since it can predict the fitness value of x without actually querying it. We may thus assume that the first query of A is either z^1 or z^2. Let $\mathcal{C}$ be the set of all sets $\{z^1, z^2\}$, where $z^1, z^2 \in \{0,1\}^n$ have Hamming distance $n/2$. In the first phase the algorithm A essentially assigns to each set $\{z^1, z^2\} \in \mathcal{C}$ either z^1 or z^2. Let us denote the corresponding function by $h_A : \mathcal{C} \to \{0,1\}^n$. With probability $1/2$, $h_A(z^1, z^2)$ is the global optimum, and with probability $1/2$ it is not.

With probability $1/2$ the algorithm enters the second phase, in which we no longer allow it to access $\{z^1, z^2\}$. The algorithm A can be randomized in this phase. Recall that the instance is taken uniformly at random, and that A samples z^1 whenever $z^2 \in \mathcal{C}_{z^1} := h_A^{-1}(z^1)$. Therefore, conditioned on seeing z^1, the global optimum is uniformly distributed in $\mathcal{C}_{z^1}$. The algorithm therefore needs an expected number of $\Omega(\mathcal{C}_{z^1})$ additional queries to find z^2, and the probability to find the optimum with $\alpha|\mathcal{C}_{z^1}|$ additional queries is at most α.

It remains to show that $\mathcal{C}_{z^1}$ is large with high probability. Let $p > 0$. Note that the average size over all z^1 (not the expectation over all instances!) of $\mathcal{C}_{z^1}$ is $E := |\mathcal{C}|/2^n = \binom{n}{n/2}/2 \sim 2^n/\sqrt{2\pi n}$. Let $D := \{z^1 \in \{0,1\}^n \mid |\mathcal{C}_{z^1}| \leq pE\}$. Then $|h_A^{-1}(D)| \leq 2^n pE \leq p|\mathcal{C}|$. Since the random instance is chosen uniformly at random from $\mathcal{F}$, the set $\{z^1, z^2\} \in \mathcal{C}$ is also uniformly at random, and with probability at least $1-p$ an instance from $\mathcal{C} \setminus h_A^{-1}(D)$ is chosen, and thus $|\mathcal{C}_{z^1}| > pE$. Thus for every $p > 0$, conditioned on entering the second phase $|\mathcal{C}_{z^1}| > pE = \Omega(p2^n/\sqrt{n})$ with probability at least $1-p$. Choosing somewhat arbitrarily $\alpha = p = n^{-1/4}$ shows that with probability at least $1/2 - o(1)$ the algorithm needs at least $\Omega(2^n/n)$ steps. This concludes the proof. $\square$

3.2 Hidden Paths

We provide another example with an exponential gap between elitist and non-elitist black-box complexities, which gives some more insight into the disadvantage of elitist algorithms. We use essentially the OneMax function, patched with a path of low fitness that leads to the global optimum. In this example, every elitist algorithm fails with high probability to find the optimum in polynomial time, since it is blind to all search points of small fitness value. Both the Monte Carlo and the Las Vegas elitist black box complexity of the problem are exponential in n, so that (unlike the example from Section 3.1) the problem cannot be easily mended by allowing restarts. On the other hand, there are memory-restricted, unary unbiased (but not elitist) algorithms that solve the problem efficiently.

For $z \in \{0,1\}^n$, let $\bar{z}$ be the bitwise complement of z, i.e., $\bar{z}_i = 1 - z_i$ for all $i \in [n]$. Let further $\mathcal{I}_\ell = \{\vec{i} = (i_1, \ldots, i_\ell) \in [n]^\ell \mid i_1, \ldots, i_\ell \text{ pairwise distinct}\}$. To each $\vec{i} \in \mathcal{I}_\ell$ and each $z^0 \in \{0,1\}^n$, we associate a path $P(z^0, \vec{i}) = (z^0, \ldots, z^\ell)$ of length ℓ as follows. For $j \in [\ell]$, let $z^j \in \{0,1\}^n$ be the search point obtained from z^{j-1} by flipping the i_j-th bit. Note that z^j differs from z^0 in exactly j bits.

Now we regard the set of all OneMax functions Om_z padded with a path of length $n/4$ starting at the minimum $z^0 := z^0(z) = \bar{z}$ and leading to the unique global maximum $z^{n/4}$. Formally, let $\mathcal{F} := \{\text{Om}_{z,\vec{i}} \mid z \in \{0,1\}^n, \vec{i} \in \mathcal{I}_{n/4}\}$, where for $\vec{i} \in \mathcal{I}_\ell$ and $P(z^0, \vec{i}) = (z^0, \ldots, z^\ell)$,

$$\text{Om}_{z,\vec{i}}(x) := \begin{cases} n + \text{Om}_z(x), & \text{if } x \notin P(z^0, \vec{i}), \\ j, & \text{if } x = z^j \text{ for } 0 \le j < \ell, \\ 2n+1, & \text{if } x = z^\ell. \end{cases}$$

Theorem 4. *The unary unbiased (1+1) memory-restricted black-box complexity of $\mathcal{F}$ is $O(n^2)$, while its (1+1) Monte Carlo (and thus, also Las Vegas) elitist black-box complexity is $2^{\Omega(n)}$, also for the non-ranking-based version of the elitist model in which full (absolute) fitness information is revealed to the algorithm.*

Proof. For the upper bound, we need to describe a memory-restricted unary unbiased black-box algorithm A that optimizes $f \in F$ in quadratic time. The algorithm proceeds as follows. While its current search point has fitness at least n, it finds the local optimum z using Randomized Local Search (RLS). This takes expected time $O(n \log n)$. From z it jumps to the starting point $z^0 = \bar{z}$ of the path $P(z, \vec{i})$. The algorithm now follows the path by using again RLS but accepting an offspring if and only if it increases the parent's fitness by exactly 1 or if the offspring's fitness is $2n+1$. In particular, in this phase the algorithm rejects any search point with fitness between n and $2n$. Since this algorithm needs time $O(n)$ to advance one step on the path, and the path has length $O(n)$, it has expected runtime $O(n^2)$.

For the lower bound we again extend the class of elitist black-box algorithms to a larger class $\mathcal{A}$ that allows to apply Yao's Principle. After an algorithm in $\mathcal{A}$ has sampled its first search point, we distinguish two cases. If the search point has fitness at most $n + n/4$, then the algorithm may access the position of the global optimum (and thus, terminate in one more step). If the first search point has fitness larger than $n + n/4$, then the algorithm may access the position of the local optimum z. Moreover, it may access a counter that tells it how many steps it has performed so far. Apart from that, it may only access (one of) the best search point(s) it has found so far, and its fitness. Then $\mathcal{A}$ is the set of all algorithms that can be implemented with this additional information. In this way, every randomized algorithm in $\mathcal{A}$ is a convex combination of deterministic ones, so that we can apply Yao's Principle. So let $A \in \mathcal{A}$ be a deterministic algorithm, and consider the uniform distribution on $\mathcal{F}$.

If the first search point has fitness $2n+1$ or at most $n+n/4$, then A is done after one query or it can terminate in at most one additional step, respectively. However, by the Chernoff bound these two events happen only with probability $e^{-\Omega(n)}$, so from now on we assume that the first search point has fitness larger than $n + n/4$. Observe that by the accessible information the algorithm can determine the OneMax value $\text{Om}_z(x)$ for all $x \in \{0,1\}^n$. In particular, for every search point of larger fitness except for z^ℓ the algorithm can predict the fitness value without querying it. On the other hand, if it queries a search point of lower fitness, then it is not allowed to keep its fitness value. Thus A cannot obtain additional information about f except by querying the optimum z^ℓ. Since $\mathcal{F}$ was chosen uniformly at random, all search points in distance $\ell = n/4$ from $z^0 = \bar{z}$ have the same probability to be the global optimum. Hence, A needs in expectation at least $\binom{n}{n/4}/2 = 2^{\Omega(n)}$ queries to find the optimum. $\square$

Remark 5. *A similar statement as the one in Theorem 4 holds also for ranking-based algorithms if we slightly increase the memory of the algorithms regarded. Indeed, there exists a unary unbiased (2+1) memory-restricted ranking-based algorithm optimizing $\mathcal{F}$ in expected $O(n^2)$ function evaluations. Regard, e.g., the algorithm that maintains throughout the second phase a search point x^1 of fitness $n+1$ and that accepts an offspring y of z^j if and only if the fitness of y is larger than that of z^j but smaller than that of x^1 (in which case $y = z^{j+1}$). Then z^ℓ is sampled (but not accepted into the population, see Remark 6) after $O(n^2)$ steps.*

On the other hand, the (2+1) elitist black-box complexity is still exponential, since with high probability the first two search points the algorithm samples have fitness $n/2 + o(n)$.

Remark 6. *As indicated in Remark 5 it can make a difference for black-box algorithms if we only require them to* sample *an optimum or whether we require the algorithm to* accept *it into the population. The algorithm described in Remark 5 does not accept the optimum when finding it. In order to achieve the same (asymptotic) black-box complexity with an algorithm that accepts the optimum, it has to be modified (and thus becomes more complicated).*

4. COMBINING UNBIASED AND ELITIST BLACK-BOX MODELS

In this section we demonstrate that apart from providing more realistic lower bounds for some function classes, the elitist black-box model is also an interesting counterpart to existing black-box models. Indeed, we show that some of the unrealistically low black-box complexities of the unbiased black-box model proposed in [22] disappear when elitist selection is required.

More specifically, we regard the unary unbiased (1+1) elitist black-box complexity[3] of JUMP functions, which (in line with [6]) we define in the following way. For a parameter k the function JUMP_k assigns to each bit string x the function value $\text{JUMP}_k(x) = \text{ONEMAX}(x)$ if $\text{ONEMAX}(x) \in \{0\} \cup \{k+1, \ldots, n-k-1\} \cup \{n\}$ and $\text{JUMP}_k(x) = 0$ otherwise. Despite the fact that all common search heuristics need $\Omega(n^{k+1})$ fitness evaluations to optimize this function, the unary unbiased black-box complexity of these functions are surprisingly low, see Table 1 for a summary of results presented in [6] ([9] for $k = 1$). Interestingly, even for extreme jump functions in which only the fitness value $n/2$ is visible and all other ONEMAX values are replaced by zero, polynomial-time unary unbiased black-box algorithms exist. It is thus interesting to see that the situation changes dramatically when the algorithms are required to be elitist, as the following theorem shows.

THEOREM 7. *For $k = 0$ the (Las Vegas and Monte Carlo) unary unbiased (1+1) elitist black-box complexity of the jump function JUMP_k is $\Theta(n \log n)$. For all $1 \le k \le n/2 - 1$ it is $\Theta(\binom{n}{k+1})$. In particular, for $k = \omega(1)$ the runtime is superpolynomial in n and for $k = \Omega(n)$ it is exponential.*

PROOF OF THE UPPER BOUND IN THEOREM 7. For any constant k the upper bound is achieved by the simple $(1+1)$ EA [15]. For general k, consider the algorithm A that produces an offspring as follows. With probability $1/3$ the offspring is a search point uniformly at random from $\{0,1\}^n$, with probability $1/3$ the algorithm flips exactly one bit (uniformly at random), and with probability $1/3$ it flips exactly $k+1$ bits (also uniformly at random). The offspring is accepted if its fitness is at least the fitness of the current search point. This algorithm finds a point of positive fitness in expected time $O(\sqrt{n})$ since it produces random search points with probability $1/3$ and each such uniform sample has ONEMAX value $n/2$ with probability $\Theta(1/\sqrt{n})$. Then with high probability it increases the fitness to $n-k-1$ in at most $O(n \log n)$ steps by one-bit flips (and possibly $(k+1)$-bit flips). Afterwards, since there are $\binom{n}{k+1}$ search points in distance $k+1$, the algorithm needs in expectation at most $3\binom{n}{k+1}$ steps to find the optimum. This proves the upper bound for the Las Vegas complexity, which in turn implies the upper bound for the Monte Carlo complexity. □

It remains to prove the lower bound. For $k = 0$ it follows from [22, Theorem 6]. For general k, as an intermediate step, we show the following general result.

THEOREM 8. *Assume that f is a function with a unique global maximum x_{opt}, and assume further that a unary unbiased (1+1) elitist black-box algorithm A is currently at a search point $x \neq x_{opt}$. Let $0 < d \le n/2$, and let* dist *denote the Hamming distance. Let*

$$S := \{x' \neq x_{opt} \mid \text{dist}\{x', x_{opt}\} \le d \text{ or } \text{dist}\{x', x_{opt}\} \ge n-d\}$$

be the distance-d neighborhood of x_{opt} and of its bitwise complement $\overline{x_{opt}}$. If all search point in S have fitness less than $f(x)$, then A needs in expectation at least $\binom{n}{d+1}$ additional queries to find the optimum. Moreover, for every $\alpha \ge 0$, the probability that it needs at most $\alpha\binom{n}{d+1}$ additional queries is at most α.

[3]That is, the complexity with respect to all (1+1) elitist black-box algorithms for which the sampling distributions in line 7 of Algorithm 1 are unbiased in the sense of Lehre and Witt [22].

PROOF. Since A is elitist, it can never accept a point in S. Therefore, in every subsequent step before finding the optimum, it will be in some search point y with distance $d' \in [d+1, n-d-1]$ from the optimum. If an unbiased mutation has some probability p to produce x_{opt} from y, then every other search point in distance d' has also probability p to be the offspring. In particular, since there are $\binom{n}{d'}$ such points, $p\binom{n}{d'}$ equals the probability that the offspring has distance d' of y, which is at most 1. Hence, at any point the probability to sample the optimum in the next step is at most $1/\binom{n}{d'} \le 1/\binom{n}{d+1}$. Therefore, A needs in expectation at least $\binom{n}{d+1}$ steps to find the optimum. Moreover, by the union bound the probability that A needs less than $\alpha\binom{n}{d+1}$ steps is at most α. □

PROOF SKETCH FOR THE LOWER BOUND IN THEOREM 7. Assume for simplicity that n is a power of 2. In order to apply Yao's Principle (Lemma 2), we allow the algorithm to remember its complete search history. We then regard a deterministic algorithm A on a uniformly chosen JUMP_k function. That is, the target string z of the ONEMAX function underlying the JUMP_k function is chosen from $\{0,1\}^n$ uniformly at random.

It is known that we can partition the hypercube $\{0,1\}^n$ into $2^n/n$ sets $S_1, \ldots, S_{2^n/n}$ of size n each such that for each $i \in [2^n/n]$ the pairwise distance between any two points in S_i is exactly $n/2$ (e.g., the cosets of the Hadamard code [2]).

Let i be the index of the set containing the target string z, i.e., $z \in S_i := \{s^1, \ldots, s^n\}$. Regardless of the jump-size k, each search point in S_i has positive fitness. Indeed, for each j either we have $s^j = z$ (in which case the fitness of s^j equals n) or the distance and thus the fitness of s^j to z equals $n/2$. Since z is chosen uniformly at random, the probability that the first search point of set S_i that the algorithm A queries is the target string z equals $1/n$. This shows that with probability at least $1 - 1/n$ the first search point with positive fitness that A queries is not the optimum. But then Theorem 8 tells us that the algorithm needs at least $\binom{n}{k+1}$ additional steps in expectation, and at least $\alpha\binom{n}{k+1}$ with probability at least α. This proves the claim. □

5. CONCLUSIONS

We have introduced elitist black-box complexity as a tool to analyze the performance of search heuristics with elitist selection rules. Several examples provide evidence that the elitist black-box complexities can give a much more realistic estimation of the runtime of typical search heuristics. We have also seen that some unrealistically low black-box complexities in the unbiased model disappear when elitist selection is enforced.

We have also introduced the concept of Monte Carlo black-box complexities and have brought to the attention of the community the fact that these can be significantly lower than the previously regarded Las Vegas complexities. In addition, it can also be significantly easier to derive bounds for the Monte Carlo black-box complexities, see [13]. Both complexity notions correspond to runtime analysis statements often seen in the evolutionary computation literature and should thus co-exist in black-box complexity research.

While we regard in this work toy-problems, it would be interesting to analyze the influence of elitist selection on the performance of algorithms in more challenging optimization problems. Our findings enliven the question for which

Model	Constant Jump $1 \leq k = \Theta(1)$	Short Jump $k = O(n^{1/2-\varepsilon})$	Long Jump $k = (1/2-\varepsilon)n$	Extreme Jump $k = n/2-1$
unary unbiased	$\Theta(n \log n)$	$\Theta(n \log n)$	$O(n^2)$	$O(n^{9/2})$
elitist unary unbiased	$\Theta(n^{k+1})$	$\Theta(\binom{n}{k+1}) = \Omega\left((n/k)^k\right)$	$\Theta(\binom{n}{k+1}) = 2^{\Theta(n)}$	$\Theta(2^n/\sqrt{n})$

Table 1: Comparison of the unary unbiased black-box complexities of Jump$_k$ with the respective (Las Vegas and Monte Carlo) elitist ones for different regimes of k.

problems non-elitist selection like tournament or so-called fitness-dependent selection can be beneficial, initial findings for which can be found in [18, 25]. Negative examples are presented in [19, 23, 24].

Acknowledgments

This research benefited from the support of the "FMJH Program Gaspard Monge in optimization and operation research", and from the support to this program from EDF.

6. REFERENCES

[1] G. Anil and R. P. Wiegand. Black-box search by elimination of fitness functions. In *FOGA'09*, pages 67–78. ACM, 2009.

[2] S. Arora and B. Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, 2009.

[3] G. Badkobeh, P. K. Lehre, and D. Sudholt. Unbiased black-box complexity of parallel search. In *PPSN'14*, volume 8672 of *LNCS*, pages 892–901. Springer, 2014.

[4] G. Badkobeh, P. K. Lehre, and D. Sudholt. Black-box complexity of parallel search with distributed populations. In *FOGA'15*. ACM, 2015. To appear.

[5] B. Doerr, C. Doerr, and F. Ebel. From black-box complexity to designing new genetic algorithms. *Theoretical Computer Science*, 567:87–104, 2015.

[6] B. Doerr, C. Doerr, and T. Kötzing. Unbiased black-box complexities of jump functions: how to cross large plateaus. In *GECCO'14*, pages 769–776. ACM, 2014.

[7] B. Doerr, C. Doerr, and T. Kötzing. The unbiased black-box complexity of partition is polynomial. *Artificial Intelligence*, 216:275–286, 2014.

[8] B. Doerr, T. Kötzing, J. Lengler, and C. Winzen. Black-box complexities of combinatorial problems. *Theoretical Computer Science*, 471:84–106, 2013.

[9] B. Doerr, T. Kötzing, and C. Winzen. Too fast unbiased black-box algorithms. In *GECCO'11*, pages 2043–2050. ACM, 2011.

[10] B. Doerr and C. Winzen. Playing Mastermind with constant-size memory. *Theory of Computing Systems*, 55:658–684, 2014.

[11] B. Doerr and C. Winzen. Ranking-based black-box complexity. *Algorithmica*, 68:571–609, 2014.

[12] B. Doerr and C. Winzen. Reducing the arity in unbiased black-box complexity. *Theoretical Computer Science*, 545:108–121, 2014.

[13] C. Doerr and J. Lengler. Onemax in black-box models with several restrictions. In *GECCO'15*. ACM, 2015. Full version: http://arxiv.org/abs/1504.02644.

[14] S. Droste, T. Jansen, K. Tinnefeld, and I. Wegener. A new framework for the valuation of algorithms for black-box optimization. In *FOGA'03*, pages 253–270. Morgan Kaufmann, 2003.

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

[16] S. Droste, T. Jansen, and I. Wegener. Upper and lower bounds for randomized search heuristics in black-box optimization. *Theory of Computing Systems*, 39:525–544, 2006.

[17] H. Fournier and O. Teytaud. Lower bounds for comparison based evolution strategies using vc-dimension and sign patterns. *Algorithmica*, 59:387–408, 2011.

[18] T. Friedrich, P. S. Oliveto, D. Sudholt, and C. Witt. Analysis of diversity-preserving mechanisms for global exploration. *Evolutionary Computation*, 17(4):455–476, 2009.

[19] E. Happ, D. Johannsen, C. Klein, and F. Neumann. Rigorous analyses of fitness-proportional selection for optimizing linear functions. In *GECCO'08*, pages 953–960. ACM, 2008.

[20] J. Jägersküpper and T. Storch. When the plus strategy outperforms the comma strategy and when not. In *FOCI'07*, pages 25–32. IEEE, 2007.

[21] T. Jansen. On the black-box complexity of example functions: The real jump function. In *FOGA'15*. ACM, 2015. To appear.

[22] P. K. Lehre and C. Witt. Black-box search by unbiased variation. *Algorithmica*, 64:623–642, 2012.

[23] F. Neumann, P. S. Oliveto, and C. Witt. Theoretical analysis of fitness-proportional selection: landscapes and efficiency. In *GECCO'09*, pages 835–842. ACM, 2009.

[24] P. S. Oliveto and C. Witt. On the runtime analysis of the simple genetic algorithm. *Theoretical Computer Science*, 545:2–19, 2014.

[25] P. S. Oliveto and C. Zarges. Analysis of diversity mechanisms for optimisation in dynamic environments with low frequencies of change. *Theoretical Computer Science*, 561, Part A:37 – 56, 2015.

[26] J. Rowe and M. Vose. Unbiased black box search algorithms. In *GECCO'11*, pages 2035–2042. ACM, 2011.

[27] O. Teytaud and S. Gelly. General lower bounds for evolutionary algorithms. In *PPSN'06*, volume 4193 of *LNCS*, pages 21–31. Springer, 2006.

[28] R. K. Ursem. Diversity-guided evolutionary algorithms. In *PPSN'02*, volume 2439 of *LNCS*, pages 462–471. Springer, 2002.

[29] M. Črepinšek, S.-H. Liu, and M. Mernik. Exploration and exploitation in evolutionary algorithms: A survey. *ACM Computing Surveys*, 45(3):35:1–35:33, 2013.

[30] A. C.-C. Yao. Probabilistic computations: Toward a unified measure of complexity. In *FOCS'77*, pages 222–227. IEEE, 1977.