Introducing Elitist Black-Box Models: When Does Elitist Behavior Weaken the Performance of Evolutionary Algorithms?

Carola Doerr
CNRS and Sorbonne Universités, UPMC Univ Paris 06, CNRS, LIP6 UMR 7606, 4 place Jussieu, 75005 Paris, France

Johannes Lengler
ETH Zürich, Department of Computer Science, Institute for Theoretical Computer Science, 8092 Zürich, Switzerland

doi:10.1162/EVCO_a_00195

Abstract

Black-box complexity theory provides lower bounds for the runtime of black-box optimizers like evolutionary algorithms and other search heuristics 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 (the relative performance of) a fixed number of the best search points sampled so far. Our elitist model thus combines features of the ranking-based and the memory-restricted black-box models with an enforced usage of truncation selection. We provide several examples for which the elitist black-box complexity is exponentially larger than that of 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 $\mathcal{F}$).

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. In simple words, the black-box complexity of a class of functions is the minimal number of function queries that is needed until, for an arbitrary member of the class, an optimal solution is queried for the first time, where the minimum is taken over all algorithms belonging to some specific class of black-box algorithms (formal definitions will be given in Section 2). The black-box models hence differ in the specifications of the algorithms under consideration. Different specifications typically yield different lower bounds for the efficiency of search heuristics. By

Manuscript received: August 27, 2015; revised: April 26, 2016 and September 23, 2016; accepted: September 27, 2016.

comparing these lower bounds, we learn how certain algorithmic choices influence the running time of evolutionary algorithms. For example, if we know that for a problem P the $\mathcal{C}$-black-box complexity is $\mathrm{BBC}_{\mathcal{C}}(P)$, while its unrestricted black-box complexity (i.e., its black-box complexity with respect to *all* black-box algorithms) is only $\mathrm{BBC}_{\mathcal{U}}(P)$, then the restrictions used to define $\mathcal{C}$ account for the discrepancy in best possible runtime.

Several models exist, each designed to analyze a different aspect of search heuristics. For example, the memory-restricted model (Doerr and Winzen, 2014a; Droste et al., 2006) helps us to understand the influence of the population size on the efficiency of the search strategy, while the ranking-based black-box model (Doerr and Winzen, 2014b; Fournier and Teytaud, 2011; Teytaud and Gelly, 2006) analyzes how much a heuristic loses by not using absolute but merely relative fitness values.

Having been introduced to the evolutionary computation community by Droste et al. (2003, 2006), black-box complexity is a young but highly active area of current research efforts (Anil and Wiegand, 2009; Badkobeh et al., 2014, 2015; Doerr et al., 2014a,b, 2013; Doerr and Winzen, 2014a,b,c; Doerr, Johannsen, et al., 2011; Fournier and Teytaud, 2011; Jansen, 2015; Lehre and Witt, 2012; Rowe and Vose, 2011; Teytaud and Gelly, 2006) (each of these papers will be mentioned below). The insights from black-box complexity studies can be used to design more efficient genetic algorithms, as the recent $(1 + (\lambda, \lambda))$ GA from Doerr et al. (2015) 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 behavior on the performance of search heuristics. We do so by enforcing that the algorithms in this model maintain a population that contains only the μ best so-far sampled individuals. Here, μ is a parameter of the model (called the *memory-* or *population-size*), while quality is measured according to increasing fitness values. Note that for population size $\mu > 1$, the μ best so-far sampled search points may have (but need not have) different fitness values. If more than μ search points of current-best fitness have been sampled, only μ of them can be stored in the population. In other words, we include in this model only those algorithms which maintain a population size of μ always, which use truncation selection in the selection for replacement step, and which base all their decisions solely on relative fitness values. While the restrictions on memory size and ranking basedness have been analyzed in the isolated models mentioned above, the combination of the two has not been studied before (with the exception of our own recent work on the OneMax function (Doerr and Lengler, 2015b)), nor has been the effect of an enforced truncation selection, despite the fact that in evolutionary computation (EC) the usage of truncation 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 emphasize that our elitist model combines three different restrictions: one on the size of the memory, one on basing all decisions only on the ranking of the search points with respect to their fitness (and not on absolute fitness values), and one on the selection for replacement step. We further note that the term "elitist" is not used consistently in the EC literature. Some subcommunities in EC call an algorithm elitist if and only if the next generation *consists* of the μ best so-far search points (e.g., Auger and Doerr (2011, page 145)—this is the most commonly applied interpretation in the theory of EAs, and is the one that we base our nomenclature on), while others speak of an elitist algorithm if and only if the next generation *contains* at least one of the best so-far solutions (Auger and Doerr, 2011, page 22), and yet another group requires an elitist algorithm to keep in the population *every* best so-far search point (so that the next population must be larger than μ if there are more than μ search points of current-best fitness, cf. Beyer et al. (2002)). Finally, a fourth notion of elitism requires that the new population consists *only* of the search point of the current-best fitness value, thus the next population must be smaller than μ if there are less than μ best so-far search points (e.g., the algorithm in Yang (2007) uses such a selection).

A short version of this work has been presented at the 2015 GECCO conference in Madrid, Spain (Doerr and Lengler, 2015a).

1.1 Previous Work

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* model (Doerr and Winzen, 2014a; Droste et al., 2006) and the *parallel* black-box model (Badkobeh et al., 2014, 2015) analyze the influence of the population size, the *unbiased* model (Doerr et al., 2013; Lehre and Witt, 2012; Rowe and Vose, 2011) 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 (Doerr and Winzen, 2014b; Fournier and Teytaud, 2011; Teytaud and Gelly, 2006), 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 the latter two 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. By providing only relative fitness values, the models aim at understanding how this worsens the performance of the algorithms, and indeed it can be shown that for some function classes the ranking-based and the comparison-based black-box complexities are larger than the unrestricted ones.

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, a strategy that is in use by many common and widely applied black-box optimization strategies such as $(\mu + \lambda)$ EAs or local hill climbers such as Randomized Local Search (RLS). On the other hand, many practical algorithms intentionally keep suboptimal solutions to enhance population diversity, or to better explore the search space (Ursem, 2002; Črepinšek et al., 2013). It has been shown that in some situations, specific elitist algorithms like RLS or the $(\mu + 1)$ EA are inferior to nonelitist algorithms (Friedrich et al., 2009; Jägersküpper and Storch, 2007; Oliveto and Zarges, 2015). In this article, we go one step further and investigate the performance of *all* elitist algorithms simultaneously.

1.2 Our Model, New Complexity Measures, and Results

We provide in this work a model to analyze the impact of elitist behavior on the runtime of black-box optimizers. In our *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 may be broken arbitrarily. For example, if more than μ search points of current-best fitness have been sampled, only (an arbitrary selection of) μ of them can be stored in the population. All other previously sampled search points are not allowed to influence the behavior of the algorithm any more. Furthermore, we do not reveal absolute fitness values to the algorithm, thus forcing it to base all its decisions on relative fitness values.

We show (Section 3) that already for quite simple function classes there can be an exponential gap between the efficiency of elitist and nonelitist black-box algorithms. As we shall see in Theorem 5, this remains true even if we regard $(1+1)$ memory-restricted unary unbiased comparison-based algorithms, which constitutes the most restrictive combination of the 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. 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 Jump_k, a classical benchmark function in evolutionary computation, is of order $\Omega(\binom{n}{k+1})$ and thus nonpolynomial for $k = \omega(1)$. In contrast, the unary unbiased black-box complexity of Jump_k is known to be polynomial even for extreme values of k (Doerr et al., 2014a).

In previous models, the black-box complexity has been defined in a *Las Vegas* manner; that is, it measures the expected number of function evaluations until the algorithm hits the optimum. On the other hand, many results in the black-box complexity literature are based on algorithms that with high (or constant) probability find the optimum 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, algorithms are not allowed to perform 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 $f : \{0,1\}^n \rightarrow \mathbb{R}$. However, generalizations to nonfinite or continuous search spaces are straightforward.

2 The Elitist Black-Box Model

The *elitist black-box model* covers all algorithms that follow the pseudocode in Algorithm 1. To describe it in a more detailed fashion, a $(\mu+\lambda)$ *elitist black-box algorithm* is initialized by sampling μ search points. We allow these search points to be sampled *adaptively*; that is, the i-th sample may depend on the ranking of the first $i-1$ search points, where, obviously, by ranking we regard the ranking induced by the fitness function f. To be very precise here, we note that two search points have the same rank if and only if they have the same fitness; that is, the search points of X with maximal f-values are rank one, the ones with second largest f-values are rank two and so on.

The *optimization phase* proceeds in rounds. In each round a $(\mu+\lambda)$ elitist black-box algorithm samples λ new search points from distributions that depend only on the current population X and the ranking of X. Note that in such an optimization step the offspring do not need to be independent of each other. Assume, for example, that we create an offspring x by random crossover; that is, we take some parents from the current population and set the entries of x by choosing (in an arbitrary way) some bit values from these parents; then it is allowed to also create another offspring y from these

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 $X \leftarrow X \setminus \{x\}$;

parents whose entries y_i in those positions i in which the parents do not agree equal $1 - x_i$. These two offspring are obviously not independent of each other. However, we do require that the offspring are created *before* any evaluation of the offspring happens. That is, the k-th offspring may *not* depend on the ranking or fitness of the first $k - 1$ offspring. (We have decided for this version as we feel that it best captures the spirit of EAs such as the $(\mu + \lambda)$ EA that can process the λ offspring in parallel.) When all λ search points have been generated, the algorithm proceeds to the *selection for replacement* step. In this step, the algorithm sorts the $\mu + \lambda$ search points according to their fitness ranking, where it is free to break ties in any way. Then the new population consists of the μ best search points according to this ordering; that is, *truncation selection* is applied.

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 nonelitist 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 offspring can be selected (comma strategies), or a nonranking-based version in which 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 nonranking-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 unbiased in the sense of Lehre and Witt (2012). 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 Doerr and Winzen (2014a) and Droste et al. (2006); 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* (i.e., 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 Doerr and Winzen (2014b). This means that the algorithm has no information about absolute fitness values, but it knows how the fitness values of the $(\mu + \lambda)$ search points compare to each other. To stress the difference between the latter two models, we remark the following: for $\mu > 1$ or $\lambda > 1$ the ranking-based black-box model provides more information than the comparison-based one as it gives a full ranking of all current search points, while in the comparison-based we always have to select two search points which are compared against each other. The ranking-based black-box complexity can thus be smaller by a logarithmic factor than the comparison-based complexity.[1]

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 nonconstant $p = p(n) = o(1)$, thus yielding high probability statements (we say that an event $\mathcal{E} = \mathcal{E}(n)$ happens *with high probability* if $\Pr[\mathcal{E}(n)] \to 1$ for $n \to \infty$).

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 Doerr et al. (2014b, 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 an 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 within the first 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

[1]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 its ranking-based black-box complexity is only $\Theta(n/\log n)$ (Doerr and Winzen, 2014b), its comparison-based complexity is $\Theta(n)$ by a straightforward application of Yao's Principle.

A on a function f the random variable describing the number of function evaluations of A until it evaluates for the first time an optimal search point of f. In this way, the Las Vegas complexity of $\mathcal{A}$ on $\mathcal{F}$ is the worst-case (over all $f \in F$) expected 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.

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 $O(T)$.

PROOF: Let R be the expected runtime of A on f. By Markov's inequality and the law of total expectation we have

$$\begin{aligned}\Pr\left[R \geq \frac{1-p_{\mathcal{E}}}{p-p_{\mathcal{E}}} \cdot T\right] &\leq \Pr\left[R \geq \frac{1-p_{\mathcal{E}}}{p-p_{\mathcal{E}}} \cdot T \,\middle|\, \neg\mathcal{E}\right] \cdot \Pr[\neg\mathcal{E}] + \Pr[\mathcal{E}] \\ &\leq \frac{p-p_{\mathcal{E}}}{1-p_{\mathcal{E}}}(1-p_{\mathcal{E}}) + p_{\mathcal{E}} = p.\end{aligned}$$

□

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 bounding the expected 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 lower bound for the expected performance of a best possible randomized algorithm for the regarded problem. In most applications a very easy distribution on the input can be chosen, often the uniform one. Formally, Yao's Principle is the following.

LEMMA 2 (Yao's Principle (Yao, 1977)): *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,*

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

where I_p denotes a random input chosen from $\mathcal{I}$ according to p, A_q a random algorithm chosen from $\mathcal{A}$ 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

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

$z \in \{0, 1\}^n$, of the well-known ONEMAX problem (see Section 3 for some background on this problem). Let A be any deterministic algorithm. Then we will show that 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 $\textsc{Om}_z$-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 $\textsc{Om}_z$ is infinite. It remains to show that this situation happens with positive probability. Assume that the first two search points that A queries are x and y. Note that A does not obtain any information from querying x, so y is independent of the fitness function. Moreover, we may assume $x \neq y$. Then there are at most 2^{n-1} search points z such that $\textsc{Om}_z(x) = \textsc{Om}_z(y)$. Moreover, by symmetry (and the uniformity of p) half of the at least 2^{n-1} remaining search points satisfy $\textsc{Om}_z(x) < \textsc{Om}_z(y)$, so A runs into an infinite loop with positive probability. Thus 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; that is, 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 algorithms where such an anomaly occurs and find the putative nonapplicability 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 (more precisely, a probability distribution over) deterministic ones. A lower bound shown for this broader class trivially applies to all elitist black-box algorithms. In particular, in a class of black-box algorithms in which every algorithm knows the number of previous queries, we may apply Yao's Principle since in such models every randomized strategy is a convex combination of deterministic strategies. The reason for that is a well-known (and completely formal) argument, which we summarize below.

The main idea is that for every (fixed) possible outcome of the random decisions of the randomized algorithm A there exists a deterministic algorithm that behaves exactly the same way. For sake of exposition, we will assume first that A flips only a single coin for each query. Then we claim that for every $N > 0$, for the first N queries A can be obtained as a convex combination of deterministic algorithms; that is, we may randomly choose a deterministic algorithm which behaves exactly like A for the first N queries. In fact, it is very easy to choose such an algorithm: we just do N coin flips in advance, and let A use the i-th coin flip for the i-th query (using that A knows the number of previous queries). Note that after fixing the coin flips, A becomes a deterministic algorithm for the first N queries. Thus we may regard the N coin flips as a way to choose a deterministic instantiation of A. The same argument carries over if we do not limit ourselves to N queries (we use an infinite number of coin flips), and if A flips more than one bit per query (we flip not just one coin for the i-th query, but rather an infinite sequence of coins).

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. In particular, the black-box complexity will still be small 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 will also show that the Las Vegas complexity of a problem can be exponentially larger than its Monte Carlo complexity.

3.1 Twin Peaks

We first describe a type of landscape for which the elitist black-box complexity is exponentially large. The following theorem captures the intuition that elitist algorithms are very bad if there are several local optima that the algorithm needs to explore in order to determine the best one of them. This remains true if we grant the algorithm access to the absolute (instead of the relative) fitness values, as we will show in Remark 4.

THEOREM 3: *Let $\varepsilon > 0$. Let $\mathcal{F}$ be a class of functions from $\{0,1\}^n$ to $\mathbb{R}$ such that for every set $\{z_1, z_2\} \subset \{0,1\}^n$ with $z_1 \neq z_2$,*

- *there is a function $f_{z_1,z_2} \in \mathcal{F}$ such that z_1 is the unique global optimum, and z_2 is the unique second-best search point of f_{z_1,z_2};*
- *$\mathcal{F}$ also contains the function f'_{z_1,z_2} that is obtained from f_{z_1,z_2} by switching the fitness of z_1 and z_2. More formally, f'_{z_1,z_2} is defined by $f'_{z_1,z_2}(z_1) = f_{z_1,z_2}(z_2)$, $f'_{z_1,z_2}(z_2) = f_{z_1,z_2}(z_1)$, and $f'_{z_1,z_2}(z) = f_{z_1,z_2}(z)$ for $z \in \{0,1\}^n \setminus \{z_1, z_2\}$.*

Then the $(1+1)$ elitist Las Vegas black-box complexity and the $(1/2-\varepsilon)$-Monte Carlo black-box complexity of $\mathcal{F}$ is $\Omega(2^n)$.

PROOF OF THEOREM 3: To give an intuition, we first give an outline of the proof that is not quite correct. Assume that a black-box algorithm encounters either f_{z_1,z_2} or f'_{z_1,z_2}. By definition of f'_{z_1,z_2}, it does not know the global optimum before querying either z_1 or z_2. It thus needs to query either z_1 or z_2 first. Assume that it queries z_1 first. Then if the algorithm is unlucky (if z_1 is not the global optimum; that is, the algorithm optimizes f'_{z_1,z_2}), the algorithm is stuck in a local optimum which it cannot leave except by sampling the optimum z_2. Due to the memory restriction the algorithm has lost any information about the objective function except possibly that z_1 is one of the two best search points. But since $f'_{z_1,z} \in \mathcal{F}$ for all $z \in \{0,1\}^n \setminus \{z_1\}$, the algorithm would then have lost any information about z_2, and would still have test $2^n - 1$ possible optima.

Unfortunately, this intuitive argument fails: after querying z_1 the algorithm does have some information about z_2, despite the severely restricted memory. For illustration, consider the following toy case. Let $n = 2$, so the search space consists of only four search points z_1, z_2, z_3, z_4. To keep the size of $\mathcal{F}$ as small as possible, and still to satisfy the second condition of the theorem, we assume that $f'_{z_i,z_j} = f_{z_j,z_i}$ for all i, j, and that $\mathcal{F}$ contains no further functions. So $\mathcal{F}$ contains exactly 12 functions, one for each ordered pair of search points. Moreover, assume that every function in $\mathcal{F}$ has value set $\{1, 2, 3, 4\}$ so that the fitness tells us whether we are in the best or second-best search point. Finally assume that $g := f_{z_2,z_1}$ has the property $g(z_2) > g(z_1) > g(z_4) > g(z_3)$, and $h := f_{z_3,z_1}$ has the property $h(z_3) > h(z_1) > h(z_4) > h(z_2)$. Consider the following algorithm A. The first query of A is always z_4. If A is at z_4, then it queries z_3, and z_1 with probability 99% and 1%, respectively. If A is at z_2 or z_3, then the next query is random. We want to understand what the algorithm "knows" when it is at z_1, and z_1 is a second-best point. In this case, the only functions to be considered are f_{z_2,z_1}, f_{z_3,z_1}, and f_{z_4,z_1}. The algorithm knows that z_4 cannot be the optimum since it always queries z_4 first. The functions f_{z_2,z_1} and f_{z_3,z_1} were a priori equally likely to be chosen. However, for the function f_{z_3,z_1} the algorithm is very unlikely to visit z_1, since from z_4 to algorithm goes directly to the optimum with probability 99%. On the other hand, for f_{z_2,z_1} the algorithm always visits z_1, since z_3 is rejected in this case. Therefore, by Bayes' theorem the a posteriori probability that the optimum is at z_2 is 99%. Hence, although there are three options z_2, z_3, z_4 for the optimum, the option z_4 is impossible, and the option z_3 is unlikely. (We remark that A is not a good algorithm because it may not always terminate; however, this can easily be fixed without affecting the argument.)

The example above shows that it is *not* true in general that A has lost all information about the search space when it enters the second-best search point. Rather it can still draw information from the order in which it typically queries search points. However, informally this information is limited to one bit, namely the (possibly probabilistic) question whether for the function f_{z_1,z_2} it first visits z_1 or z_2 first. Therefore, the algorithm cannot gain much, and the intuitive argument outlined at the beginning still works approximately, as we will show below.

To turn the above 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. Assume the algorithm has to optimize f_{z_1,z_2} or f'_{z_1,z_2}. We call the time until the algorithm queries for the first time z_1 or z_2 the "first phase," while we call the remaining time the "second phase." Since we are not too much interested in the time that the algorithm spends in the first phase, we simply give away to the algorithm the set $\{f_{z_1,z_2}, f'_{z_1,z_2}\}$. That is, we give the algorithm complete information about the functions f_{z_1,z_2} and f'_{z_1,z_2}, but do not tell the algorithm which one of the two is the one to be optimized. Since the two function coincide everywhere except for z_1 and z_2, during the first phase the algorithm knows everything about the objective function except which of the two points z_1 or z_2 is the optimum. We also give the algorithm access to unlimited memory throughout this phase. During this phase every randomized algorithm is a convex combination of deterministic ones. So we may use Yao's Principle, choose a probability distribution on $\mathcal{F}$, and restrict ourselves to an algorithm A that is deterministic in the first phase. For the probability distribution on $\mathcal{F}$, we choose a set $\{z_1, z_2\}$ of two n-bit strings uniformly at random, and then we pick either f_{z_1,z_2} or f'_{z_1,z_2}, each with probability 1/2.

Note that in the first phase the algorithm does not gain any additional information by querying any search point $z \notin \{z_1, z_2\}$ since it can predict the fitness value of z 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 $z_1, z_2 \in \{0, 1\}^n$ and $z_1 \neq z_2$. 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 anything but the current search point and possibly its fitness. For the sake of exposition, we first consider the case that the algorithm may not access the fitness, and describe afterwards how to change the argument otherwise. The algorithm A can be randomized in this second 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 $\mathcal{C}_{z_1}$. Since A does not have any additional memory in this phase, every subsequent query has probability at most $1/|\mathcal{C}_{z_1}|$ to be the optimum, independent of any previous queries. Hence, A needs in expectation at least $|\mathcal{C}_{z_1}|$ 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 by the union bound.

It remains to show that $\mathcal{C}_{z_1}$ is large with high probability. Let $p > 0$. Since the sets $\mathcal{C}_z$ form a partition of $\mathcal{C}$, and since there are 2^n such sets, the average size[2] of the $\mathcal{C}_z$ is $E := |\mathcal{C}|/2^n = (2^n - 1)/2$. Let $D_p := \{z \in \{0, 1\}^n \mid |\mathcal{C}_z| \leq pE\}$. Then $|h_A^{-1}(D_p)| \leq 2^n pE \leq p|\mathcal{C}|$. 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 $\mathcal{C} \setminus h_A^{-1}(D_p)$ is chosen, and thus $|\mathcal{C}_{z_1}| > pE$. Thus for every $p > 0$, conditioned on entering the second phase we have $|\mathcal{C}_{z_1}| > pE = \Omega(p2^n)$ with probability at least $1 - p$. Choosing $\alpha = p = \varepsilon/2$ shows that with probability at least $1/2 - \varepsilon$ the algorithm needs at least $\Omega(2^n)$ steps. This concludes the proof. □

REMARK 4: Theorem 3 essentially also holds if we allow the algorithms to access absolute fitness values. More precisely, let $\mathcal{F}$ be a class of functions as in Theorem 3, and let $V(\mathcal{F}) := \max\{f(z) \mid f \in F, z \in \{0, 1\}^n, z \text{ is not a global maximum of } f\}$ be the set of all second-best fitness values. If $V(\mathcal{F})$ has subexponential size, then the $(1 + 1)$ elitist Las Vegas black-box complexity and the $(1/2 - \varepsilon)$-Monte Carlo black-box complexity of $\mathcal{F}$ remain exponential even if the algorithms have access to the absolute fitness values.

PROOF: The same proof as for Theorem 3 still works, only that for every $a \in V(\mathcal{F})$ we let $\mathcal{C}_{z_1,a} := \{z_2 \in \mathcal{C}_{z_1} \mid f_{z_1,z_2}(z_1) = a\}$. This partitions $\mathcal{C}_{z_1}$ into $V(\mathcal{F})$ subsets, and since $|V(\mathcal{F})| = 2^{o(n)}$, on average these sets are still exponentially large. The theorem now follows in the same way as before, with the sets $\mathcal{C}_{z_1}$ replaced by $\mathcal{C}_{z_1, f(z_1)}$. □

The Double OneMax Problem Theorem 3 provides us with landscapes that are very hard for elitist algorithms. We now give a more concrete example, the class of *double* ONEMAX *functions*. This class is of the type as described in Theorem 3, but at the same time it is easy for a very simple nonelitist algorithm, namely a variant of RLS using restarts (cf. Algorithm 3). The basis for double ONEMAX 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 bit string. 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 expected runtime for any function from the generalized

[2] This is not the same as the expected size of $\mathcal{C}_{z_1}$ if we pick an instance f_{z_1,z_2} uniformly at random. Since some sets $\mathcal{C}_z$ correspond to more instances f_{z_1,z_2} than others, choosing the instance uniformly at random does not yield the uniform distribution over the $\mathcal{C}_z$. So it is in general *not* true that for a uniformly random instance the resulting $\mathcal{C}_z$ has expected size $(2^n - 1)/2$.

Algorithm 2: Randomized local search for maximizing $f\colon \{0,1\}^n \to \mathbb{R}$.

Initialization: Sample $x \in \{0,1\}^n$ uniformly at random and query $f(x)$;
Optimization: for $t = 1, 2, 3, \ldots$ **do**
  Choose $j \in [n]$ uniformly at random;
  Obtain y by flipping the j-th bit of x, and query $f(y)$; //mutation step
  if $f(y) \geq f(x)$ **then** $x \leftarrow y$; //selection step;

OneMax function class OneMax $:= \{\text{Om}_z \mid z \in \{0,1\}^n\}$, where Om_z is defined by (2). We call z, the unique global optimum of function Om_z, the *target string* of Om_z.

OneMax is one of the best-understood problems in the theory of evolutionary computation, and serves as a showcase also in many publications on black-box complexity. Most notably, it is known that the unrestricted black-box complexity of OneMax is $\Theta(n/\log n)$ (Anil and Wiegand, 2009; Droste et al., 2006; Erdös and Rényi, 1963), and that this bound holds also in the ranking-based (Doerr and Winzen, 2014b) and the $(1+1)$ memory-restricted (Doerr and Winzen, 2014a) black-box models. The unary unbiased black-box complexity (cf. Section 4 for a brief explanation of this model) of OneMax is $\Theta(n \log n)$ (Lehre and Witt, 2012), its binary unbiased black-box complexity is linear (Doerr, Johannsen, et al., 2011), and its k-ary unbiased black-box complexity is $\Theta(n/k)$ (Doerr and Winzen, 2014c). Finally, the $(1+1)$ elitist Monte Carlo black-box complexity of OneMax is $\Theta(n)$ (Doerr and Lengler, 2015b).

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 our subsequent proofs, we give its pseudocode in Algorithm 2. RLS is initialized with a uniform sample x. In each iteration one bit position $j \in [n] := \{1, \ldots, n\}$ 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, where unbiased refers to the notion of unbiasedness defined in Lehre and Witt (2012, Section 3.2).[3]

We are now ready to define the double onemax functions. For two different strings $z_1, z_2 \in \{0,1\}^n$, let

$$\text{Om}_{z_1,z_2}(x) := \begin{cases} \max\{\text{Om}_{z_1}(x), \text{Om}_{z_2}(x)\}, & \text{if } x \neq z_1, \\ n+1, & \text{otherwise.} \end{cases}$$

The unique global optimum of this function is z_1, and z_2 is the unique second best search point. For all $x \notin \{z_1, z_2\}$ the fitness $\text{Om}_{z_1,z_2}(x)$ equals $\text{Om}_{z_2,z_1}(x)$. 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 $\mathcal{F} := \{\text{Om}_{z_1,z_2} \mid z_1, z_2 \in \{0,1\}^n, z_1 \neq z_2\}$ and show the following.

Theorem 5: *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*

[3]For the interested reader we mention that Lehre and Witt define unbiasedness only for variation operators working on the n-dimensional hypercube. An extension to other search spaces has been provided in Doerr et al. (2013), while a very general definition for "unbiasedness" can be found in Rowe and Vose (2011).

Algorithm 3: Randomized local search with random restarts.

Initialization: Sample $x \in \{0,1\}^n$ uniformly at random and query $f(x)$;
Optimization: for $t = 1, 2, 3, \ldots$ **do**
  With probability $1/(2n \ln n)$ sample $y \in \{0,1\}^n$ uniformly at random and replace x by y;
  else
    Choose $j \in [n]$ uniformly at random;
    Obtain y by flipping the j-th bit of x, and query $f(y)$; //mutation step
    if $f(y) \geq f(x)$ **then** $x \leftarrow y$; //selection step;

$O(n \log n)$, while the $(1+1)$ elitist Las Vegas black-box complexity of $\mathcal{F}$ and its $(1/2 - \varepsilon)$-Monte Carlo black-box complexity are $\Omega(2^n)$ even if we allow the algorithms to access absolute fitness values.

Proof of Theorem 5: The class $\mathcal{F}$ satisfies the conditions from Theorem 3, so the lower bound for the $(1+1)$ elitist Las Vegas black-box complexity of $\mathcal{F}$ follows immediately from Theorem 3 and Remark 4. For the upper bound, consider the random local search algorithm (RLS) with random restarts as given by Algorithm 3. This algorithm is initialized like RLS. The only difference to RLS (Algorithm 2) is that during the optimization process, instead of mutating the current best search point, it may restart completely by drawing a point y uniformly at random from $\{0, 1\}^n$ and replacing the current best solution x by y regardless of their fitness values. 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 with respect to whether or not $\textsc{Om}_{z_1}(x) \geq \textsc{Om}_{z_2}(x)$). This is at least as large as the progress probability for OneMax. Since it is well-known that with high probability RLS finds the optimum of OneMax within, say, $2n \ln n$ steps (Auger and Doerr, 2011, Theorem 1.23), with high probability it finds either z_1 or z_2 in this time. Therefore, if no restarts happen in $2n \ln n$ steps (which is true with constant probability), then with high probability Algorithm 3 also finds either z_1 or z_2 in this time. Note that the search space is 2-vertex transitive; that is, there is an automorphism of the search space that maps z_1 to z_2 and vice versa. By definition of $\textsc{Om}_{z_1,z_2}(x)$, the same automorphism maps $\textsc{Om}_{z_1,z_2}(x)$ to $\textsc{Om}_{z_2,z_1}(x)$. Hence, since RLS with restarts is an unbiased algorithm, it will reach z_1 before z_2 with probability $1/2$, and vice versa. Thus, when the algorithm queries either z_1 or z_2, then it finds the global optimum with probability $1/2$. Summarizing, after each restart, the algorithm has at least a constant probability to find the global optimum in the next $2n \ln n$ steps. This proves both upper bounds in Theorem 5. □

3.2 Hidden Paths

We provide another example with an exponential gap between elitist and nonelitist 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; that is, $\bar{z}_i = 1 - z_i$ 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, we choose $\ell := n/4$ and set $\mathcal{F} := \{\text{Om}_{z,\vec{i}} \mid z \in \{0,1\}^n, \vec{i} \in \mathcal{I}_\ell\}$, where for $\vec{i} \in \mathcal{I}_\ell$ and $P(z^0, \vec{i}) = (z^0, \ldots, z^\ell)$,

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

Each function in $\mathcal{F}$ has a clearly distinguished path leading to the optimum, which can be easily tracked even by unbiased memory-restricted algorithms. However, the path is of low fitness, so an elitist algorithm will lose track quickly (in fact, it will not even find the path in the first place). Formally, we prove the following theorem.

Theorem 6: *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 nonranking-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, and after the query it is in the same state as before. Either way, the algorithm can predict in which state it will be if the query does not hit the optimum, so 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. □

REMARK 7: A similar statement as the one in Theorem 6 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 8) after $O(n^2)$ steps.

On the other hand, the (2+1) elitist black-box complexity is still exponential, since with probability at least $1-o(1)$ (in fact, at least $1-e^{-\Omega(n)}$) the ONEMAX values of the first two search points are at least $n/4$.

REMARK 8: As indicated in Remark 7 it can make a crucial difference for (nonelitist) black-box algorithms if we only require them to *sample* an optimum or whether we require the algorithm to *accept* it into the population. For example, the algorithm described in Remark 7 does not accept the optimum when finding it.

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 Lehre and Witt (2012) disappear when elitist selection is required.

More specifically, we regard the unary unbiased $(1+1)$ elitist black-box complexity[4] of JUMP functions, which we define in the following way (this definition is in line

[4]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 (2012). In brief, a variation operator is *k-ary unbiased* if it generates new search points only from at most k old ones by changing them in a way that does not discriminate between the bit positions $\{1, 2, \ldots, n\}$ nor between the bit values 0 and 1. For several problems the unary unbiased black-box model provides quite reasonable bounds for the performance of typically investigated evolutionary algorithms. However, it is also known that even in this model some problems have a complexity that is much smaller than the performance of any general-purpose black-box algorithm, cf. Doerr et al. (2014b) for an NP-hard problem having a small polynomial unary unbiased black-box complexity.

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.

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

with Doerr et al. (2014a), but deviates from Droste et al. (2002); most notably, in our definition the k and not the $k-1$ fitness values before n are blanked out. See Jansen (2015) for an alternative way to generalize the JUMP function class. 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 mutation-based 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 Doerr et al. (2014a); Doerr, Kötzing, et al. (2011) 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 9: *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 $1 \leq k \leq n/2 - 1$ it is $\Theta(\binom{n}{k+1})$. In particular, for $k = \omega(1)$ the black-box complexity is superpolynomial in n and for $k = \Omega(n)$ it is $2^{\Omega(n)}$.*

PROOF OF THE UPPER BOUND IN THEOREM 9: For any constant k the upper bound is achieved by the simple $(1 + 1)$ EA (Droste et al., 2002). 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. We claim that this algorithm finds a point of positive fitness in expected time $O(\sqrt{n})$. Indeed, it produces random search points with probability $1/3$, and each such uniform sample has ONEMAX value $n/2$ with probability $\binom{n}{n/2}/2^n = \Theta(1/\sqrt{n})$ by Stirling's formula. In summary, in each step the algorithm produces queries a search point of fitness $n/2$ with probability $\Theta(1/\sqrt{n})$, so the expected time until a search point of positive fitness is queried is at most $O(\sqrt{n})$, as claimed. Once such a search point is found, by a coupon collector argument 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 Lehre and Witt (2012, Theorem 6). For general k, as an intermediate step, we show the following general result.

THEOREM 10: *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 $0 < d \leq n/2$, and let dist denote the Hamming distance. Let*

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

be the distance-d neighborhood of x_{opt} and of its bitwise complement $\overline{x_{opt}}$. If all search points 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 $\alpha \geq 0$, the probability that it needs at most $\alpha\binom{n}{d+1}$ additional queries is at most α.

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 $1/\binom{n}{d'} \leq 1/\binom{n}{d+1}$. 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 at most $\alpha\binom{n}{d+1}$ steps is at most α. □

PROOF FOR THE LOWER BOUND IN THEOREM 9: We give the proof only in the case that n is a power of 2, which is less technical. Consider a unary unbiased $(1+1)$ elitist black-box algorithm A. Let $\mathcal{E} = \mathcal{E}_A$ be the event that the first search point of *strictly positive* fitness that A queries is the optimum. We claim that $\Pr[\mathcal{E}] \leq 1/n$. Before we prove the claim, we discuss how it implies the theorem. Conditioned on $\neg\mathcal{E}$, Theorem 10 tells us that A needs at least $\binom{n}{k+1}$ additional steps in expectation, and at most $\alpha\binom{n}{k+1}$ with probability at most α. Thus, the probability to find the optimum in at most $\alpha\binom{n}{k+1}$ steps is at most $\alpha + \Pr[\mathcal{E}] \leq \alpha + 1/n$. Rephrasing this statement, for every (constant) $0 < p < 1$ the algorithm needs more than $(1 - p - 1/n)\binom{n}{k+1} = \Omega(\binom{n}{k+1})$ steps with probability at least p. This proves the lower bound on the Monte Carlo complexity, which in turn implies the lower bound on the Las Vegas complexity by Markov's inequality.

So it remains to show that $\Pr[\mathcal{E}_A] \leq 1/n$. In fact, we will show that this is true for *every* (unrestricted) black-box algorithm A. Note that such unrestricted algorithms are in particular not memory restricted, so by Yao's Principle (Lemma 2) it suffices to proof the lower bound for all *deterministic* back-box algorithms on random input. So let A be such a deterministic algorithm. We regard 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. For ease of terminology, we say that A *wins* if the first search point of positive fitness that A queries is the optimum, that A *loses* otherwise, and that A *terminates* with the i-th query if A either wins or loses with the i-th queries.

Consider the following sequence of search points. Let z^1 be the first query of A, and for $i > 1$ let z^i be the search point that A queries in round i if the previous search points $z^1, \ldots, z^{i-1}$ all had fitness 0. We may assume that the queries z^i are all different from each other, and in this case the sequence $z^1, \ldots, z^{2^n}$ forms a permutation of the search space that determines A. (In fact, z^i may be ill-defined for large i because it can happen that A terminates with probability 1 with the first $i-1$ queries. For example, if $k = 1$, then there are only $2n + 1$ search points of fitness 0, so A terminates with probability 1 with the first $2n + 1$ queries. In this case, for consistency of notation we fill up the sequence in an arbitrary way, with the queries $z^i, \ldots, z^{2^n}$ being irrelevant for the question whether A wins or loses.) With this notation, the event $\mathcal{E}$ can equivalently be phrased as the event that the optimum z is not the first search point of positive fitness in the list $z^1, \ldots, z^{2^n}$.

If n is a power of 2 then it is well known that one can partition the hypercube $\{0, 1\}^n$ into $2^n/n$ sets $S_1, \ldots, S_{2^n/n}$ of size n 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 Walsh-Hadamard code as described in Section 17.5.1 of Arora and Barak, 2009). 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 z is the first search point of set S_i to appear in the sequence $z^1, \ldots, z^{2^n}$ equals $1/n$. On the other hand, if z is *not* the first search point of S_i, then this implies the event $\mathcal{E}$. Therefore, $\Pr[\mathcal{E}] \geq 1 - 1/n$, as required. □

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 expected 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 Doerr and Lengler, 2015b). 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 problems nonelitist selection like tournament or so-called fitness-dependent selection can be beneficial, initial findings for which can be found in Friedrich et al. (2009) and Oliveto and Zarges (2015). Negative examples are presented in Happ et al. (2008); Neumann et al. (2009); Oliveto and Witt (2014).

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 (Électricité de France).

References

Anil, G., and Wiegand, R. P. (2009). Black-box search by elimination of fitness functions. In *Proceedings of Foundations of Genetic Algorithms*, pp. 67–78.

Arora, S., and Barak, B. (2009). *Computational complexity: A modern approach*. Cambridge: Cambridge University Press.

Auger, A., and Doerr, B. (2011). *Theory of randomized search heuristics*. Singapore: World Scientific.

Badkobeh, G., Lehre, P. K., and Sudholt, D. (2014). Unbiased black-box complexity of parallel search. In *Proceedings of Parallel Problem Solving from Nature*, pp. 892–901. Lecture Notes in Computer Science, vol. 8672.

Badkobeh, G., Lehre, P. K., and Sudholt, D. (2015). Black-box complexity of parallel search with distributed populations. In *Proceedings of Foundations of Genetic Algorithms*, pp. 3–15.

Beyer, H.-G., Brucherseifer, E., Jakob, W., Pohlheim, H., Sendhoff, B., and To, T. B. (2002). Evolutionary algorithms—Terms and definitions (glossary, version e-1.2). Retrieved from http://ls11-www.cs.uni-dortmund.de/~beyer/EA-glossary/node35.html

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

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

Doerr, B., Doerr, C., and Kötzing, T. (2014a). Unbiased black-box complexities of jump functions: How to cross large plateaus. In *Proceedings of Genetic and Evolutionary Computation Conference (GECCO'14)*, pp. 769–776.

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

Doerr, B., Johannsen, D., Kötzing, T., Lehre, P. K., Wagner, M., and Winzen, C. (2011). Faster black-box algorithms through higher arity operators. In *Proceedings of Foundations of Genetic Algorithms*, pp. 163–172.

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

Doerr, B., Kötzing, T., and Winzen, C. (2011). Too fast unbiased black-box algorithms. In *Proceedings of Genetic and Evolutionary Computation Conference (GECCO'11)*, pp. 2043–2050.

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

Doerr, B., and Winzen, C. (2014b). Ranking-based black-box complexity. *Algorithmica*, 68:571–609.

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

Doerr, C., and Lengler, J. (2015a). Elitist black-box models: Analyzing the impact of elitist selection on the performance of evolutionary algorithms. In *Proceedings of Genetic and Evolutionary Computation Conference (GECCO'15)*, pp. 839–846.

Doerr, C., and Lengler, J. (2015b). OneMax in black-box models with several restrictions. In *Proceedings of Genetic and Evolutionary Computation Conference (GECCO'15)*, pp. 1431–1438.

Droste, S., Jansen, T., Tinnefeld, K., and Wegener, I. (2003). A new framework for the valuation of algorithms for black-box optimization. In *Proceedings of Foundations of Genetic Algorithms*, pp. 253–270.

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

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

Erdös, P., and Rényi, A. (1963). On two problems of information theory. *Magyar Tudományos Akadémia Matematikai Kutató Intézet Közleményei*, 8:229–243.

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

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

Happ, E., Johannsen, D., Klein, C., and Neumann, F. (2008). Rigorous analyses of fitness-proportional selection for optimizing linear functions. In *Proceedings of Genetic and Evolutionary Computation Conference (GECCO'08)*, pp. 953–960.

Jägersküpper, J., and Storch, T. (2007). When the plus strategy outperforms the comma strategy and when not. In *Proceedings of the IEEE Symposium on Foundations of Computational Intelligence*, pp. 25–32.

Jansen, T. (2015). On the black-box complexity of example functions: The real jump function. In *Proceedings of Foundations of Genetic Algorithms*, pp. 16–24.

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

Neumann, F., Oliveto, P. S., and Witt, C. (2009). Theoretical analysis of fitness-proportional selection: Landscapes and efficiency. In *Proceedings of Genetic and Evolutionary Computation Conference (GECCO'09)*, pp. 835–842.

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

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

Rowe, J., and Vose, M. (2011). Unbiased black box search algorithms. In *Proceedings of Genetic and Evolutionary Computation Conference (GECCO'11)*, pp. 2035–2042.

Teytaud, O., and Gelly, S. (2006). General lower bounds for evolutionary algorithms. In *Proceedings of Parallel Problem Solving from Nature*, pp. 21–31. Lecture Notes in Computer Science, vol. 4193.

Ursem, R. K. (2002). Diversity-guided evolutionary algorithms. In *Proceedings of Parallel Problem Solving from Nature*, pp. 462–471. Lecture Notes in Computer Science, vol. 2439.

Yang, S. (2007). Genetic algorithms with elitism-based immigrants for changing optimization problems. In *Proceedings of Applications of Evolutionary Computing, EvoWorkshops 2007*, pp. 627–636. Lecture Notes in Computer Science, vol. 4448.

Yao, A. C.-C. (1977). Probabilistic computations: Toward a unified measure of complexity. In *Proceedings of Foundations of Computer Science*, pp. 222–227.