Artificial Intelligence 216 (2014) 275–286

Contents lists available at ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

# The unbiased black-box complexity of partition is polynomial

Benjamin Doerr [a], Carola Doerr [b,c,*], Timo Kötzing [d]

[a] *École Polytechnique, Palaiseau, France*
[b] *Sorbonne Universités, UPMC Univ Paris 06, UMR 7606, LIP6, Paris, France*
[c] *CNRS, UMR 7606, LIP6, Paris, France*
[d] *Friedrich-Schiller-Universität Jena, Germany*

ARTICLE INFO

*Article history:*
Received 4 September 2013
Received in revised form 20 June 2014
Accepted 24 July 2014
Available online 30 July 2014

*Keywords:*
Heuristic search
Run time analysis
Evolutionary computation

ABSTRACT

Unbiased black-box complexity was introduced as a refined complexity model for randomized search heuristics (Lehre and Witt (2012) [24]). For several problems, this notion avoids the unrealistically low complexity results given by the classical model of Droste et al. (2006) [10].

We show that for some problems the unbiased black-box complexity remains artificially small. More precisely, for two different formulations of an $\mathcal{NP}$-hard subclass of the well-known PARTITION problem, we give mutation-only unbiased black-box algorithms having complexity $O(n \log n)$. This indicates that also the unary unbiased black-box complexity does not give a complete picture of the true difficulty of this problem for randomized search heuristics.



## 1. Introduction

Complexity theory aims at determining the difficulty of computational problems. In classical theoretical computer science, the fruitful interplay between complexity theory, aiming at proving that a certain effort is necessary to solve a problem, and theory of algorithms, giving an algorithmic solution for a problem and showing that it can be solved with a certain computational effort, was a driving force to develop the field.

Developing a similarly thorough theory for heuristic search methods is at the heart of the theory of randomized search heuristics community. The latter has been steadily growing in the last twenty years. Many tight or near-tight run time analyses for various problems and algorithms exist, see, e.g., the recent textbooks [26,4,21]. In contrast to this, the complexity theory of randomized search heuristics is still in its infancy.

*A complexity theory for randomized search heuristics* Applications of search heuristics typically do not consider the problem as explicitly given, in contrast to traditional optimization algorithms which operate on a problem instance containing all relevant information. Instead, a search heuristic supposes access to the problem instance via an oracle/as a black box and the heuristic can learn about the concrete instance at hand only by learning the function value of the search points it generates. Search heuristics are therefore called *black-box* optimization algorithms. The reasons for this restricted access are manifold: sometimes no explicit representation of the problem exists (a function evaluation might correspond to a real world experiment), sometimes we do not want to fully exploit the given representation (e.g., due to its size or complexity). The efficiency of heuristic search methods is measured by the number of function evaluations until an optimal search point is evaluated for the first time, or, depending on the context, until a solution of a certain quality has been found. This is

* Corresponding author.

http://dx.doi.org/10.1016/j.artint.2014.07.009
0004-3702/

very different from classical complexity notions (e.g., the Turing model of computation), which assume that the algorithm is fully aware of the complete problem instance, and where efficiency of an algorithm is measured by counting the number of arithmetic operations that it performs given the input data.

As the run time (also referred to as the optimization time) of a search heuristic is measured by the number of function evaluations, this should be reflected in a corresponding complexity model. This is what black-box complexity models are developed for. They try to give a reasonable estimate of how easily a problem can be optimized by typical search heuristic methods. A useful black-box complexity notion thus provides lower bounds for a sufficiently large class of black-box search methods. In the simplest black-box complexity model, the so-called *unrestricted black-box complexity*, one just counts the number of function evaluations that are necessary to solve a problem, no single restriction is made on how the search points are being generated. The unrestricted black-box complexity gives thus a lower bound for *all* black-box algorithms, i.e., all algorithms that do not exploit knowledge about the problem instance other than those obtained from the function evaluations—this class in particular contains all stochastic hill-climbing algorithms but also bio-inspired search heuristics like evolutionary algorithms, physics-inspired algorithms like simulated annealing, and many other classes of search heuristics. Hence, the unrestricted black-box complexity is a very general notion, independently studied in a number of different scientific communities (see Section 2.3 for more details).

While theoretically very pleasing, it quickly turned out that the class of all black-box algorithms is possibly too wide to deduce powerful statements for the most commonly used randomized search heuristics. Already in the seminal work of Droste, Jansen, Tinnefeld, and Wegener [9] (see [10] for the journal version), it was observed that there are black-box algorithms solving the MaxClique problem (the optimization version of the $\mathcal{NP}$-complete decision problem Clique) using a polynomial number of function evaluations.[1]

To increase the practical relevance of black-box complexity theory, a number of more restrictive models have been developed. A promising direction is the so-called *unbiased black-box complexity model* by Lehre and Witt in [24], restricting the class of admitted black-box optimization algorithms in a natural way which still includes a large class of commonly used randomized search heuristics. In this model, all solution candidates are required to be obtained by variation operators. These variation operators must be *unbiased*, that is, treat the bit positions and the bit entries 0 and 1 in an unbiased way (see Section 2 for a precise definition). The unbiased black-box model admits a notion of arity in a natural way: A $k$-ary unbiased black-box algorithm is one that employs only variation operators that take up to $k$ arguments. This allows for talking about mutation-only algorithms (*unary* algorithms, i.e., having arity one) or crossover-based algorithms (having arity at least two) from a complexity-theoretic perspective.

*Known results* For several function classes the unbiased black-box complexity model leads to more realistic complexities. While in the unrestricted model any function class consisting of a single function has a black-box complexity of one, in the unbiased model more function evaluations are needed to generate an optimal solution from applying the variation operators. For example, the mutation-only black-box complexity of any class of functions having a unique global optimum is $\Omega(n \log n)$ [24]. The (permutation-invariant) LeadingOnes function class, which is also one of the classic test problems in the theory of randomized search heuristic community, has a mutation-only black-box complexity of $\Theta(n^2)$ [24], matching the run time of standard randomized search heuristics.[2] The unrestricted black-box complexity of this LeadingOnes problem is known to be of order $\Theta(n \log \log n)$ [1]. Hence, for both the unimodal and the LeadingOnes test problems, the unary unbiased black-box model leads to much better complexity estimates than the unrestricted one.

When higher-arity variation operators are used, i.e., when the algorithm may combine two or more search points to generate a new one, smaller, but still not completely unrealistic complexities were observed in [8]. For the classic test problem OneMax, which assigns to each bit string the number of ones in it, the unary unbiased black-box complexity is $\Omega(n \log n)$ by the above mentioned lower bound, but it drops to $O(n)$ for unbiased black-box algorithms of arity two.[3] For LeadingOnes the binary unbiased complexity is shown to be $O(n \log n)$ (as opposed to the $\Omega(n^2)$ bound for the unary unbiased complexity by Lehre and Witt [24] mentioned above). For larger arity $k \leq \log_2 n$, the $k$-ary unbiased black-box complexity of OneMax is $O(n/k)$ as shown in [15]. For LeadingOnes, it is known [13] that 3-ary operators reduce the unbiased black-box complexity to $O(n \log n / \log \log n)$. In the light of [1], it seems likely that larger arities again yield further improvements, but no proof in this direction exists.

[1] This does, of course, not provide an answer to whether or not $\mathcal{P}$ equals $\mathcal{NP}$—as outlined above we do not count arithmetic operations, but just function evaluations. An algorithm may perform a super-polynomial number of arithmetic operations between any two function evaluations; see Section 2.3 for a lightweight discussion on how black-box complexity relates to classic complexity theory and for a detailed presentation of an algorithm solving the MaxClique problem in $\Theta(n^2)$ function evaluations.

[2] For the sake of completeness we give the definition of this problem. For every permutation $\pi$ of the bit positions $\{1, \dots, n\}$ and every length-$n$ bit string $z$ the leadingones function $f_{z,\pi}(\cdot)$ is defined by setting, for each bit string $x \in \{0,1\}^n$, $f_{z,\pi}(x) := \max\{i \in [0..n] \mid \forall j \leq i : z_{\pi(j)} = x_{\pi(j)}\}$; i.e., $f_{z,\pi}(x)$ is the length of the longest common prefix of $x$ and $z$ with respect to $\pi$. The function class LeadingOnes is the collection $\{f_{z,\pi}(\cdot) \mid z, \pi\}$ of all such leadingones functions.

[3] The OneMax problem may seem strange, since it is clear that the unique global optimum is the all-ones string $(1, \dots, 1)$. However, when regarded in the unbiased black-box model, this function is equivalent to any function $f_z(\cdot) : \{0,1\}^n \to \{0, 1, \dots, n\}, x \mapsto |\{i \mid x_i = z_i\}|$ counting the number of bit positions in which the evaluated string $x$ agrees with the *target string* $z$. The unrestricted black-box complexity of the class $\{f_z \mid z \in \{0,1\}^n\}$ of generalized OneMax functions is known to be $\Theta(n/\log n)$ by a classic result of Erdős and Rényi [16].

*Our result* The above-mentioned results show that the (unary) unbiased black-box model for several classic test problems gives complexity estimates much closer to the performance of commonly used randomized search heuristics. We show that this observation does not extend in general to difficult combinatorial optimization problems. We demonstrate an $\mathcal{NP}$-hard subclass of the classic combinatorial optimization problem Partition which has a small polynomial unbiased black-box complexity, even if the arity of the variation operators is restricted to one. More precisely, we consider the subclass $\textsc{Partition}_{\neq}$ of all Partition instances with pairwise different weights. For two natural formulations of this problem (one using a signed and one using an unsigned objective function, see Sections 3.1 and 3.2) we prove that there exists a unary unbiased black-box algorithm that solves any of such instances using only $O(n \log n)$ function evaluations. This result shows that there are problems for which no efficient optimization algorithm exists (making the common assumption that the complexity classes $\mathcal{P}$ and $\mathcal{NP}$ are not identical), but that have a (small) polynomial unbiased black-box complexity.

This paper is based on the conference paper [12].

## 2. Preliminaries

In this section we first introduce the notation used in this paper, followed by a formal definition of the unrestricted and the unbiased black-box models and a brief discussion of the black-box complexity concept.

### 2.1. Notation

The positive integers are denoted by $\mathbb{N}$. For any $k \in \mathbb{N}$, we abbreviate $[k] := \{1, \dots, k\}$. Analogously, we define $[0..k] := [k] \cup \{0\}$.

For a bit string $x = x_1 \cdots x_n \in \{0,1\}^n$ we denote by $\bar{x}$ the bit-wise complement of $x$ (i.e., for all $i \in [n]$ we have $\bar{x}_i = 1 - x_i$). The bit-wise exclusive-OR is denoted by $\oplus$. We say that $y$ is created from $x$ by flipping the $i$th bit in $x$ to define $y$ as $x \oplus e_i$, where $e_i$ denotes the $i$th unit vector. For any bit string $x$ we let $\textsc{OneMax}(x) = |x|_1$ denote the number of 1s in $x$ (also known as the *Hamming-weight of* $x$). Let $|x|_0 := n - |x|_1$ denote the number of zeros in $x$.

For any set $S$ we denote by $2^S$ the power set of $S$, i.e., the set of all subsets of $S$. For any set of pseudo-Boolean functions $C$ and any function $f : \mathbb{R} \to \mathbb{R}$ we let $f(C) = \{f \circ g \mid g \in C\}$.

For $n \in \mathbb{N}$, we let $S_n$ be the set of all permutations of $[n]$. For $\sigma \in S_n$ and $x \in \{0,1\}^n$ we abbreviate $\sigma(x) := x_{\sigma(1)} \cdots x_{\sigma(n)}$.

Lastly, with ln we denote the natural logarithm to base $e := \exp(1)$.

### 2.2. Unrestricted and unbiased black-box model

A usual way to measure the *complexity of a problem* is to measure the *performance of the best algorithm* out of some class of algorithms (e.g., all those algorithms which can be implemented on a Turing machine [18,20]) on the (for this algorithm) most difficult problem instance. As we would like to measure the complexity of a problem's *optimizability by randomized search heuristics*, we restrict the class of permissible algorithms to those which obtain information about the problem only by learning the objective values of possible solutions ("search points"). Thus the objective function in this setting is given as an oracle or as a *black-box*. Using this oracle, the algorithm may query the objective value of any solution; such a query does only return this search point's function value but no other information about the problem instance.

Here in this work we will be concerned only with so-called *pseudo-Boolean functions*, i.e., real-valued objective functions defined on the set $\{0,1\}^n$ of bit strings of length $n$. This is motivated by the fact that many randomized search heuristics, in particular evolutionary algorithms, use such a representation. Black-box complexity notions are meaningful also in other search domains and objective spaces, but for the sake of clarity, we restrict our definitions to the pseudo-Boolean settings. Results and models for more general search spaces can be found in [27,11].

Naturally, we do allow that the algorithms use random decisions. It follows from the black-box concept that the only type of action the algorithm may perform is, based on the objective values learned so far, deciding on a probability distribution on $\{0,1\}^n$, sampling a search point $x \in \{0,1\}^n$ according to this distribution, and querying its function value (often referred to as "fitness" in the evolutionary computation community) from the oracle. This leads to the scheme of Algorithm 1, which we call an *unrestricted black-box algorithm*.

As the performance measure of black-box algorithms we take the number of queries to the oracle performed by the algorithm until it first queries an optimal solution. We call this the *run time*, or *optimization time*, of the black-box algorithm. This is justified by the observation that, in typical applications of randomized search heuristics, evaluating the function value of a search point is more costly than the generation of a new search point. Since we mainly talk about randomized algorithms, we are interested in the *expected* number of queries.

We can now follow the usual approach in complexity theory. Let $\mathcal{F}$ be a class of pseudo-Boolean functions. The *complexity* of an algorithm $A$ for $\mathcal{F}$ is the maximum expected run time of $A$ on a function $f \in \mathcal{F}$ (worst-case run time). The complexity of $\mathcal{F}$ with respect to a class $\mathcal{A}$ of algorithms is the minimum ("best") complexity among all $A \in \mathcal{A}$ for $\mathcal{F}$. The *unrestricted black-box complexity* of $\mathcal{F}$ is the complexity of $\mathcal{F}$ with respect to the class of all (unrestricted) black-box algorithms. This is the black-box complexity as defined by Droste, Jansen, Tinnefeld, and Wegener [9,10].

Obviously, the class of all black-box algorithms is very powerful. For example, for any function class $\mathcal{F} = \{f\}$ consisting of one single function, the unrestricted black-box complexity of $\mathcal{F}$ is 1—the algorithm that simply queries an optimal solution

**Algorithm 1:** Scheme of an unrestricted black-box algorithm.

**Initialization:** Sample $x^{(0)}$ according to some probability distribution $p^{(0)}$ on $\{0,1\}^n$. Query $f(x^{(0)})$.
**Optimization: for** $t = 1, 2, 3, \ldots$ ***until*** *termination condition met* **do**
  Depending on $((x^{(0)}, f(x^{(0)})), \ldots, (x^{(t-1)}, f(x^{(t-1)})))$ choose a probability distribution $p^{(t)}$ on $\{0,1\}^n$.
  Sample $x^{(t)}$ according to $p^{(t)}$, and query $f(x^{(t)})$.

**Algorithm 2:** Scheme of a $k$-ary unbiased black-box algorithm.

**Initialization:** Sample $x^{(0)} \in \{0,1\}^n$ uniformly at random and query $f(x^{(0)})$.
**Optimization: for** $t = 1, 2, 3, \ldots$ ***until*** *termination condition met* **do**
  Depending on $(f(x^{(0)}), \ldots, f(x^{(t-1)}))$ choose up to $k$ indices $i_1, \ldots, i_k \in [0..t-1]$ and a $k$-ary unbiased distribution $D(\cdot \mid x^{(i_1)}, \ldots, x^{(i_k)})$.
  Sample $x^{(t)}$ according to $D(\cdot \mid x^{(i_1)}, \ldots, x^{(i_k)})$ and query $f(x^{(t)})$.

of $f$ at the first opportunity verifies this bound. Also, there are black-box algorithms solving MaxClique using a polynomial number of queries in expectation [10]; see Section 2.3 for a brief discussion of this algorithm.

These drawbacks of the unrestricted black-box model inspired Lehre and Witt [24] to introduce a more restrictive black-box model, where algorithms may generate new solution candidates only from random or previously generated search points and only by using *unbiased* variation operators. Still this model includes most of the commonly studied search heuristics, such as many $(\mu + \lambda)$ and $(\mu, \lambda)$ evolutionary algorithms (EAs), simulated annealing, the Metropolis algorithm, and the Randomized Local Search algorithm.

**Definition 1.** For all $k \in \mathbb{N}$, a *$k$-ary unbiased distribution* $(D(\cdot \mid y^{(1)}, \ldots, y^{(k)}))_{y^{(1)},\ldots,y^{(k)} \in \{0,1\}^n}$ is a family of probability distributions over $\{0,1\}^n$ such that for all *inputs* $y^{(1)}, \ldots, y^{(k)} \in \{0,1\}^n$ the following two conditions hold.

$$\text{(i) } \forall x, z \in \{0,1\}^n: \quad D\left(x \mid y^{(1)}, \ldots, y^{(k)}\right) = D\left(x \oplus z \mid y^{(1)} \oplus z, \ldots, y^{(k)} \oplus z\right);$$

$$\text{(ii) } \forall x \in \{0,1\}^n \, \forall \sigma \in S_n: \quad D\left(x \mid y^{(1)}, \ldots, y^{(k)}\right) = D\left(\sigma(x) \mid \sigma\left(y^{(1)}\right), \ldots, \sigma\left(y^{(k)}\right)\right).$$

We refer to the first condition as *$\oplus$-invariance* and to the second as *permutation-invariance*. An operator sampling from a $k$-ary unbiased distribution is called a *$k$-ary unbiased variation operator*.

Note that the only 0-ary unbiased distribution over $\{0,1\}^n$ is the uniform distribution. 1-ary, also called *unary* operators are sometimes referred to as mutation operators, in particular in the field of evolutionary computation. 2-ary, also called *binary* operators are often referred to as crossover operators. If we allow arbitrary arity, we call the corresponding model the $*$-ary unbiased black-box model.

$k$-ary unbiased black-box algorithms can now be described via the scheme of Algorithm 2. The *$k$-ary unbiased black-box complexity* of some class of functions $\mathcal{F}$ is the complexity of $\mathcal{F}$ with respect to all $k$-ary unbiased black-box algorithms.

Note that, for all $k \leq \ell$, each $k$-ary unbiased black-box algorithm is contained in the $\ell$-ary unbiased black-box model. For any set $C$ of pseudo-Boolean functions we write $\mathrm{UBB}_k(C)$ to denote the $k$-ary unbiased black-box complexity of $C$.

As mentioned in the introduction, Lehre and Witt [24] proved, among other results, that all functions with a single global optimum have a unary unbiased black-box complexity of $\Omega(n \log n)$. For several standard test problems this bound is met by different unary randomized search heuristics, such as the $(1+1)$ EA or the Randomized Local Search algorithm. Recall that, as pointed out above, the unrestricted black-box complexity of any single such function is 1. For results on higher arity models refer to [8,15].

Rather than bounding the expected run time of an algorithm, it is sometimes easier to show that it solves the given problem with good probability in some fixed number of iterations. If we are only interested in asymptotic black-box complexities, the following remark allows us to use such statements for computing upper bounds.

**Remark 2.** Suppose for a problem $P$ there exists a black-box algorithm $A$ that, with constant success probability, solves $P$ in $s$ iterations (that is, queries an optimal solution within $s$ queries). Then the black-box complexity of $P$ is at most $O(s)$.

**Proof.** Let $c$ be an upper bound for the failure probability of algorithm $A$ after $s$ iterations. We let $A'$ be the algorithm which performs independent runs of $s$ iterations of $A$ indefinitely one after the other. If $X_i$ denotes the indicator variable for the event that the $i$th independent run of $A$ is successful (i.e., computes an optimum), then $\Pr[X_i = 1] \geq 1 - c$. Clearly, $Y := \min\{k \in \mathbb{N} \mid X_k = 1\}$ is a geometric random variable with success probability at least $1 - c$. Hence, $\mathrm{E}[Y] = (1-c)^{-1}$, i.e., the expected number of independent runs of $A$ until success is at most $(1-c)^{-1}$. Thus, we can optimize $P$ in an expected number of at most $(1-c)^{-1}s$ iterations. Since $c$ is constant, the claim follows. □

### 2.3. Discussion of the black-box complexity concept

In this section, we briefly discuss the differences between classical complexity models and the black-box models introduced above. The reader only interested in the result on Partition can skip this section without loss.

What distinguishes black-box complexity from classical complexity is the fact that in the black-box setting

- the algorithms have no access to the problem instance other than by sampling search points and learning their function values, and that
- the cost measure is the number of such function evaluations that are needed to optimize a problem.

Recall that in classical complexity settings the problem instances are typically given as a *white box*, and instead of function evaluations, one counts the number of arithmetic operations that are needed to compute an optimal solution (or the answer to questions like satisfiability of the instance etc.). The motivation for the usage of the black-box model in evolutionary computation is that they are typically applied to problems with very large or very complex descriptions. Instead of analyzing the particular problem instance at hand, the idea is to learn about it by evaluating solution candidates. Since such evaluations are typically the most costly part of an evolutionary algorithm, the cost of an algorithm is measured by the number of such evaluations.

The black-box setting is a standard notion much beyond the evolutionary computation community. Black-box complexity is also referred to as *query complexity* in classic computer science [2,19,23]. Very similar notions are used in learning theory [3,5] and numerical integration theory (information-based complexity, see [25]).

In this short discussion, we would like to emphasize the fact that a polynomial black-box complexity of a problem does not contradict standard complexity assumptions like $\mathcal{P} \neq \mathcal{NP}$. To illustrate this, let us consider the already mentioned MaxClique problem.

The optimization variant of the well-known $\mathcal{NP}$-complete decision problem Clique is the following task: given a graph $G = (V, E)$, find a subset $W \subseteq V$ of maximal size such that $W$ forms a *clique*, i.e., such that all nodes in $W$ are mutually connected. We can model the MaxClique problem in the black-box setting by assigning to each subset $W$ of $V$ the objective value $f(W) := |W|$ if $W$ is a clique, and $f(W) := 0$ otherwise. Optimizing MaxClique then corresponds in a natural way to maximizing $f$. To model $f$ as a pseudo-Boolean function, enumerate the vertices in $V$ and identify a bit string $x \in \{0,1\}^n$ with the set of vertices for which the corresponding entry in $x$ is one.

As has already been pointed out in [10], the unrestricted black-box complexity of MaxClique is at most $\binom{n}{2} + 1$. An algorithm achieving this complexity does the following. In the first $\binom{n}{2}$ iterations, it queries the function values of all possible subsets of size two (i.e., all bit strings with exactly two ones). This corresponds to asking whether or not an edge between the two queried search points exists. Once this has been done for all $\binom{n}{2}$ possible edges, the algorithm has full knowledge about $G$. It can thus compute *offline*, i.e., without any further function evaluations, a clique of maximal size. This can be done, for example, by a brute force algorithm. The algorithm then terminates by asking in the $\left(\binom{n}{2}+1\right)$st iteration the search point that corresponds to the computed optimal solution. Clearly, it is not known how or if the offline computation can be done in polynomial (CPU) time.

## 3. Partition

The fact that optimization versions of $\mathcal{NP}$-hard problems (like the MaxClique problem mentioned in Section 2.3) can be solved efficiently in the unrestricted black-box model is one of the main criticism received by it.

In this section we prove that in the unbiased black-box model, too, there are $\mathcal{NP}$-hard problems whose optimization version have a small polynomial black-box complexity. We consider the Partition problem, which is a well known, and probably one of the most famous, $\mathcal{NP}$-hard problem, cf. [22,18]. Given a multiset $\mathcal{I}$ of positive integers ("weights"), the decision version of Partition asks whether or not it is possible to split the set into two disjoint subsets $\mathcal{I} = \mathcal{I}_0 \dot\cup \mathcal{I}_1$ such that $\sum_{w \in \mathcal{I}_0} w = \sum_{w \in \mathcal{I}_1} w$. The corresponding optimization variant of Partition asks to find a partition $(\mathcal{I}_0, \mathcal{I}_1)$ of $\mathcal{I}$ such that the difference $|\sum_{w \in \mathcal{I}_0} w - \sum_{w \in \mathcal{I}_1} w|$ is minimized. To be more precise, we consider here an $\mathcal{NP}$-hard subclass of Partition, which will be described below.

Partition is also one of the few $\mathcal{NP}$-hard problems for which theoretical investigations of randomized search heuristics exist. In fact, it is known that Partition permits heuristics which solve many instances of the problem in a polynomial number of function evaluations. For example, Frenk and Kan [17] showed that the greedy approach converges to optimality almost surely for reasonably chosen random instances. Furthermore, greedy approaches are known to deliver in polynomial number of queries solutions of good approximation quality. Witt [28] has shown that both the Randomized Local Search algorithm and the $(1+1)$ EA need at most $O(n^2)$ iterations until they reach for the first time a solution of approximation quality $4/3$. More results for the performance of greedy heuristics on Partition can be found in [6] and [28].

Despite these positive results on the performance of greedy strategies on random instances and their approximation quality, Partition is an $\mathcal{NP}$-hard problem, and it is thus widely believed that there is no algorithm solving it in polynomial time. As we shall demonstrate below, we show that the difficulty of Partition is not sufficiently captured by the unary unbiased black-box model. In fact, it is possible to solve Partition using only a small polynomial number of queries to the problem instance.

As mentioned, we consider an $\mathcal{NP}$-hard subclass of Partition. In fact, it is not difficult to see that Partition remains $\mathcal{NP}$-hard if we restrict the problem to instances with pairwise different weights. The proof is routine, and we state it here only for the sake of completeness.

**Lemma 3** *(Folklore).* Partition *remains $\mathcal{NP}$-hard when restricted to instances $\mathcal{I}$ with $v \neq w$ for all $v, w \in \mathcal{I}$.*

**Proof.** Let $\mathcal{I}$ be an instance of the general problem class Partition (i.e., $\mathcal{I}$ may contain multiple instances of the same integer). We create a new instance $\mathcal{I}'$ from $\mathcal{I}$ by the procedure described below. $\mathcal{I}'$ will have pairwise different weights only and we will prove that an optimal partition for $\mathcal{I}'$ immediately yields an optimal partition for the original input $\mathcal{I}$.

To construct $\mathcal{I}'$, we first (arbitrarily) enumerate the values in $\mathcal{I}$ by $\varphi$, i.e., $\varphi : \mathcal{I} \to [n]$ is a bijection. We then set $c := n^3$ and we let $\mathcal{I}' := \{cw + \varphi(w) \mid w \in \mathcal{I}\} \cup [n]$. By construction, all integers in $\mathcal{I}'$ do have different weights.

Let $(\mathcal{I}'_0, \mathcal{I}'_1)$ be an optimal partition for $\mathcal{I}'$. We set $\mathcal{I}_0 := \{(w - \varphi(w))/c \mid w \in \mathcal{I}'_0 \backslash [n]\}$ and $\mathcal{I}_1 := \{(w - \varphi(w))/c \mid w \in \mathcal{I}'_1 \backslash [n]\}$. We use contraposition to show that $(\mathcal{I}_0, \mathcal{I}_1)$ is an optimal solution for $\mathcal{I}$. More precisely, we show that any partition of $\mathcal{I}$ which is better than $(\mathcal{I}_0, \mathcal{I}_1)$ gives rise to a partition of $\mathcal{I}'$ which is better than $(\mathcal{I}'_0, \mathcal{I}'_1)$, contradicting the choice of $(\mathcal{I}'_0, \mathcal{I}'_1)$. Thus, if $(\mathcal{I}'_0, \mathcal{I}'_1)$ is optimal, so is $(\mathcal{I}_0, \mathcal{I}_1)$.

To prove the claim, assume that there exists a solution $(O_0, O_1)$ for $\mathcal{I}$ with $|\sum_{w \in O_0} w - \sum_{w \in O_1} w| < |\sum_{w \in \mathcal{I}_0} w - \sum_{w \in \mathcal{I}_1} w|$. We set $O'_0 := \{c \cdot w + \varphi(w) \mid w \in O_0\} \cup \{\varphi(w) \mid w \in O_1\}$ and $O'_1 := \{c \cdot w + \varphi(w) \mid w \in O_1\} \cup \{\varphi(w) \mid w \in O_0\}$. Note that $c(\sum_{w \in O_0} w - \sum_{w \in O_1} w) = \sum_{w \in O'_0} w - \sum_{w \in O'_1} w$.

We assume without loss of generality that $\sum_{w \in \mathcal{I}_0} w - \sum_{w \in \mathcal{I}_1} w > 0$. Then $\sum_{w \in \mathcal{I}'_0} w - \sum_{w \in \mathcal{I}'_1} w > 0$ since $c(\sum_{w \in \mathcal{I}_0} w - \sum_{w \in \mathcal{I}_1} w) > 0$, $\sum_{w \in \mathcal{I}_0} \varphi(w) - \sum_{w \in \mathcal{I}_1} \varphi(w) + \sum_{w \in \mathcal{I}'_0 \cap [n]} w - \sum_{w \in \mathcal{I}'_1 \cap [n]} w > \sum_{w \in \mathcal{I}_0} \varphi(w) - \sum_{w \in \mathcal{I}_1} \varphi(w) - \sum_{i \in [n]} i > -(n^2 + n) > -c$ (by definition of $c$ and assuming $n \geq 2$) and, thus,

$$\begin{aligned}\sum_{w \in \mathcal{I}'_0} w - \sum_{w \in \mathcal{I}'_1} w &= c\left(\sum_{w \in \mathcal{I}_0} w - \sum_{w \in \mathcal{I}_1} w\right) + \sum_{w \in \mathcal{I}_0} \varphi(w) - \sum_{w \in \mathcal{I}_1} \varphi(w) + \sum_{w \in \mathcal{I}'_0 \cap [n]} w - \sum_{w \in \mathcal{I}'_1 \cap [n]} w \\ &> c - c = 0.\end{aligned}$$

Similarly we obtain

$$\begin{aligned}\left|\sum_{w \in \mathcal{I}'_0} w - \sum_{w \in \mathcal{I}'_1} w\right| &> c\left(\sum_{w \in \mathcal{I}_0} w - \sum_{w \in \mathcal{I}_1} w\right) - c = c\left(\sum_{w \in \mathcal{I}_0} w - \sum_{w \in \mathcal{I}_1} w - 1\right) \\ &\geq c\left|\sum_{w \in O_0} w - \sum_{w \in O_1} w\right| = \left|\sum_{w \in O'_0} w - \sum_{w \in O'_1} w\right|,\end{aligned}$$

contradicting the optimality of $(\mathcal{I}'_0, \mathcal{I}'_1)$ for $\mathcal{I}'$ (which implies $|\sum_{w \in \mathcal{I}'_0} w - \sum_{w \in \mathcal{I}'_1} w| \leq |\sum_{w \in O'_0} w - \sum_{w \in O'_1} w|$). □

In the following, let Partition$_{\neq}$ be the subclass of Partition instances with pairwise different weights.

There is no one best way of how to consider Partition$_{\neq}$ as an optimization problem. For two different models of Partition$_{\neq}$, different with respect to the objective function, we show (Theorems 4 and 6) that the unary unbiased black-box complexity is $O(n \log n)$. That is, we give a unary unbiased black-box algorithm which solves any instance of Partition$_{\neq}$ in $O(n \log n)$ queries.

In Section 3.1 we consider a signed objective function where we learn from the function values which of the sums $\sum_{w \in I_0} w$ and $\sum_{w \in I_1} w$ is the larger one.

In Section 3.2 we then consider an unsigned objective function. This gives us, a priori, less information than the signed case. However, we are still able to prove the same asymptotic bound. This second objective function is probably the more natural one but note that the key arguments for our upper bound are essentially the same.

### 3.1. The signed objective function

As mentioned, we first consider a signed objective function for Partition$_{\neq}$. To this end, we define the sets $\mathcal{F}_{\mathcal{I}} := \{(\mathcal{I}_0, \mathcal{I}_1) \in 2^{\mathcal{I}} \times 2^{\mathcal{I}} \mid \mathcal{I}_0 \dot\cup \mathcal{I}_1 = \mathcal{I}\}$ of *feasible solutions* for instance $\mathcal{I}$. The signed objective function to measures the quality of the queried solutions is defined as

$$f^*_{\mathcal{I}} : \mathcal{F} \to \mathbb{Z}, \quad (\mathcal{I}_0, \mathcal{I}_1) \mapsto \sum_{w \in \mathcal{I}_0} w - \sum_{w \in \mathcal{I}_1} w.$$

Note that we aim at minimizing the absolute value $|f^*_{\mathcal{I}}|$.

Next we describe how we model Partition as a pseudo-Boolean problem. We fix as enumeration $\sigma : \mathcal{I} \to [n]$ the ordering of the elements in $\mathcal{I}$, i.e., $\sigma(v) < \sigma(w)$ for all $v, w \in \mathcal{I}$ with $v < w$. (All arguments that follow below could be easily

**Algorithm 3:** Unary unbiased black-box algorithm for $\textsc{Partition}_{\neq}$ with the signed objective function

1 **Initialization:**
2 Sample $x^{(0)} \leftarrow \texttt{uniform}()$. Query $f(x^{(0)})$;
3 Initialize $t \leftarrow 0$ and , $\mathcal{I}'_0, \mathcal{I}'_1, \mathcal{W}_0 = \emptyset$;
4 **Learning the integers:**
5 **while** $|\mathcal{W}_t| < n$ **do**
6 $t \leftarrow t + 1$;
7 Sample $x^{(t)} \leftarrow \texttt{RLS}(x^{(0)})$. Query $f(x^{(t)})$;
8 Update $\mathcal{W}_t \leftarrow \mathcal{W}_{t-1} \cup \{|f(x^{(0)}) - f(x^{(t)})|/2\}$;
9 **if** $f(x^{(0)}) > f(x^{(t)})$ **then**
10 $\mathcal{I}'_0 \leftarrow \mathcal{I}'_0 \cup \{|f(x^{(0)}) - f(x^{(t)})|/2\}$;
11 **else** $\mathcal{I}'_1 \leftarrow \mathcal{I}'_1 \cup \{|f(x^{(0)}) - f(x^{(t)})|/2\}$;
12 ;
13 **Optimization:**
14 Offline compute an optimal solution $(\mathcal{O}_0, \mathcal{O}_1)$ and set
$\mathcal{M} \leftarrow \{w \in \mathcal{O}_0 \mid w \notin \mathcal{I}'_0\} \cup \{w \in \mathcal{O}_1 \mid w \notin \mathcal{I}'_1\}$, the set of integers that need to be moved;
15 Set $z \leftarrow x^{(0)}$;
16 **while** $|\mathcal{M}| > 0$ **do**
17 Sample $y \leftarrow \texttt{RLS}(z)$. Query $f(y)$;
18 **if** $w := |f(y) - f(z)|/2 \in \mathcal{M}$ **then**
19 ;
20 $z \leftarrow y$ and $\mathcal{M} \leftarrow \mathcal{M} \setminus \{w\}$;

carried out using an arbitrary enumeration, but for the sake of readability we restrict ourselves to considering as $\sigma$ the natural ordering.) For any $x \in \{0,1\}^n$ let $\mathcal{I}_0(x) := \{w \in \mathcal{I} \mid x_{\sigma(w)} = 0\}$ and, accordingly, $\mathcal{I}_1(x) := \{w \in \mathcal{I} \mid x_{\sigma(w)} = 1\}$. Note that $\{0,1\}^n \to \mathcal{F}_{\mathcal{I}}, x \mapsto (\mathcal{I}_0(x), \mathcal{I}_1(x))$ is a bijection between $\{0,1\}^n$ and the original search space $\mathcal{F}_{\mathcal{I}}$. It is therefore natural to consider as objective function the function $f_{\mathcal{I}}$ which is defined by

$$f_{\mathcal{I}} : \{0,1\}^n \to \mathbb{Z}, \quad x \mapsto \sum_{i \in [n], x_i = 0} \sigma^{-1}(i) - \sum_{i \in [n], x_i = 1} \sigma^{-1}(i).$$

**Theorem 4.** *The unary unbiased black-box complexity of* $\textsc{Partition}_{\neq}$ *modeled via the signed objective functions* $f_{\mathcal{I}}$ *is* $O(n \log n)$, *where* $n := |\mathcal{I}|$ *denotes the size of the input set* $\mathcal{I}$.

Interestingly, the algorithm certifying Theorem 4 requires only two different variation operators, namely $\texttt{uniform}()$, which samples a bit string $x \in \{0,1\}^n$ uniformly at random and $\texttt{RLS}(\cdot)$ (randomized local search) which, given some $x \in \{0,1\}^n$, creates from $x$ a new bit string $y \in \{0,1\}^n$ by flipping exactly one bit in $x$, the bit position being chosen uniformly at random. The following is straightforward to verify from the definition of unbiased variation operators.

**Remark 5.** $\texttt{uniform}()$ is a (0-ary) unbiased variation operator. $\texttt{RLS}(\cdot)$ is a unary unbiased variation operator.

**Proof of Theorem 4.** We show that Algorithm 3 is an unbiased algorithm that, for any instance $\mathcal{I}$ of $\textsc{Partition}_{\neq}$, needs an expected $O(|\mathcal{I}| \log |\mathcal{I}|)$ function evaluations to compute a partition $(\mathcal{O}_0, \mathcal{O}_1) \in 2^{\mathcal{I}} \times 2^{\mathcal{I}}$ such that $|\sum_{w \in \mathcal{O}_0} w - \sum_{w \in \mathcal{O}_1} w|$ is minimized. Since it only employs the two variation operators $\texttt{uniform}()$ and $\texttt{RLS}(\cdot)$, it is an unbiased black-box algorithm of arity one.

Fix a problem instance $\mathcal{I}$ of $\textsc{Partition}_{\neq}$, let $n := |\mathcal{I}|$ denote its size, and abbreviate $f := f_{\mathcal{I}}$.

The algorithm works as follows. It starts with a uniformly sampled solution $x^{(0)}$. In a first phase, the algorithm learns all $n$ different weights of the problem instance. To this end, it samples from $x^{(0)}$ new search points $x^{(t)}$ which differ from $x^{(0)}$ in exactly one position. The steps in lines 9 to 11 are needed to remember which weights are in the same equivalence class of the partition induced by $x^{(0)}$. Note that they do not require any additional queries.

After an expected number of $(1 + o(1))n \ln n$ iterations, we have learned the weights of the problem instance as follows. First note that in the $t$th iteration of the algorithm, the weight of the flipped bit is $|f(x^{(0)}) - f(x^{(t)})|/2$. Therefore, let $\mathcal{W}_t := \{|f(x^{(0)}) - f(x^{(s)})|/2 \mid s \in [t]\}$. By a coupon collector argument (cf., e.g., Theorem 1.21 in [4]) the expected number of queries until we have flipped each bit position of $x^{(0)}$ at least once is $(1 + o(1))n \ln n$. Thus, we can expect that we need $t^* \in (1 + o(1))n \ln n$ queries until $\mathcal{W}_{t^*} = \mathcal{I}$. That is, we can assume to have learned all $n$ different weights in $\mathcal{I}$ in $(1 + o(1))n \ln n$ queries.

Knowing the problem instance $\mathcal{I}$ we can compute an optimal partition $(\mathcal{O}_0, \mathcal{O}_1)$ for $\mathcal{I}$ *offline*, i.e., we do not need to query any further search points for this step. The computation can be done, e.g., by applying the brute force algorithm which compares all $2^n$ possible solutions. All we need to do now is to create a representation of $(\mathcal{O}_0, \mathcal{O}_1)$ via unbiased variation operators of arity at most 1.

To this end let us define $\mathcal{I}'_0(x^{(0)}) := \{|f(x^{(0)}) - f(x^{(s)})|/2 \mid s \in [t^*], f(x^{(0)}) > f(x^{(s)})\}$ and, accordingly, $\mathcal{I}'_1(x^{(0)}) := \{|f(x^{(0)}) - f(x^{(s)})|/2 \mid s \in [t^*], f(x^{(0)}) < f(x^{(s)})\}$. It is easily verified that $x^{(0)}$ is a binary representation of the partition $(\mathcal{I}'_0(x^{(0)}), \mathcal{I}'_1(x^{(0)}))$.

To create $(\mathcal{O}_0, \mathcal{O}_1)$ we set $\mathcal{M} := \{w \in \mathcal{O}_0 \mid w \notin \mathcal{I}'_0(x^{(0)})\} \cup \{w \in \mathcal{O}_1 \mid w \notin \mathcal{I}'_1(x^{(0)})\}$, the set of all weights that, in order to generate the optimal solution $(\mathcal{O}_0, \mathcal{O}_1)$, need to be moved from one of the sets $\mathcal{I}'_0(x^{(0)}), \mathcal{I}'_1(x^{(0)})$ to the other one.

In the optimization phase we do the following. In each iteration we create a new solution $y$ from the current solution $z$ by flipping exactly one bit of $z$. If $w := |f(y) - f(z)|/2 \in \mathcal{M}$, we update $z \leftarrow y$ and $\mathcal{M} \leftarrow \mathcal{M} \setminus \{w\}$.

By the same coupon collector argument as above we can expect that after $(1 + o(1))n \ln n$ such one bit flips we have flipped each bit position $i \in [n]$ at least once. That is, after an expected number of $(1 + o(1))n \ln n$ queries, we have $\mathcal{M} = \emptyset$ and we have thus created $(\mathcal{O}_0, \mathcal{O}_1)$.

This shows how to optimize an arbitrary instance $\mathcal{I}$ of the PARTITION$_{\neq}$ problem modeled via the signed objective function in an expected number of $2(1 + o(1))n \ln n = O(n \log n)$ queries by an unbiased algorithm of arity one. □

### 3.2. The unsigned objective function

One might dislike the fact that in the proof of Theorem 4 we neither minimize nor maximize $f_{\mathcal{I}}$ itself but only its absolute value $|f_{\mathcal{I}}|$. However, we can achieve the same asymptotic optimization complexity as in the statement of Theorem 4 if we only allow the latter, unsigned objective function. Although the algorithm itself does not become more difficult to define, the proof of correctness is more technical. The difficulty for the analysis stems from the fact that, given two bit strings $x$ and $y$ which differ in only one bit position, we cannot unambiguously learn from the corresponding function values $|f_{\mathcal{I}}(x)|$ and $|f_{\mathcal{I}}(y)|$ the weight of the flipped bit, cf. Remark 7. This results in a slightly more complex procedure to learn the different weights.

**Theorem 6.** *The unary unbiased black-box complexity* PARTITION$_{\neq}$ *with respect to* $|f_{\mathcal{I}}|$ *is* $O(n \log n)$*, where* $n := |\mathcal{I}|$ *denotes the size of the input set* $\mathcal{I}$*.*

For a clearer presentation of the proof, we defer some technical elements used in the proof of Theorem 6 to lemmas that will be presented after the proof of the main theorem.

**Proof of Theorem 6.** By Remark 2 it suffices to show that there exists an algorithm that, for an arbitrary instance $\mathcal{I}$ of PARTITION$_{\neq}$, has at least constant success probability to compute an optimal search point within $O(|\mathcal{I}| \log |\mathcal{I}|)$ function evaluations. We present a unary unbiased algorithm, Algorithm 4 that does so even with high probability (w.h.p.), that is, with probability $1 - o(1)$.

For the remainder of the proof, we fix again an instance $\mathcal{I}$ of PARTITION$_{\neq}$, let $n := |\mathcal{I}|$, and we abbreviate $f := |f_{\mathcal{I}}|$, where $f_{\mathcal{I}}$ is defined as in Section 3.1.

For readability purposes we introduce the following notation, which—this is important to note—are a priori not identifiable for the algorithm. Using the notation from Section 3.1, each $x \in \{0,1\}^n$ clearly corresponds to a partition $(\mathcal{I}_0(x), \mathcal{I}_1(x)) \in \mathcal{F}_{\mathcal{I}}$. We set $\mathcal{S}_0(x) := \sum_{w \in \mathcal{I}_0(x)} w$ and $\mathcal{S}_1(x) := \sum_{w \in \mathcal{I}_1(x)} w$, the corresponding sums of the weights in the equivalence classes of that partition. Let $\mathcal{I}_{\max}(x) := \mathcal{I}_0(x)$ if $\mathcal{S}_0(x) \geq \mathcal{S}_1(x)$ and let $\mathcal{I}_{\max}(x) = \mathcal{I}_1(x)$ otherwise. We call $\mathcal{I}_{\max}(x)$ the "heavier" bin and we call the other one the "lighter" bin. Lastly, let $w_{\max} = \max \mathcal{I}$ be the largest weight appearing in instance $\mathcal{I}$.

The general approach of Algorithm 4 is the following. First we produce a string which represents a solution where all weights are in the same class of the partition, i.e., at the end of this phase we have $\mathcal{I}_{\max}(x) = \mathcal{I}$. With high probability this can be achieved with $4n \ln n$ queries. Next, we perform $2n \ln n$ `RLS` steps (i.e., random one-bit flips). Through this we learn all $n$ different weights in $\mathcal{I}$ w.h.p. After that, we compute an optimal solution offline. A representation of this solution can be generated in another $3n \ln n$ iterations w.h.p.

If in any iteration of Algorithm 4 we have constructed a solution $s$ with $f(s) = 0$ we are obviously done and do not need to run the algorithm any further. Therefore, we assume in the following that for all search points $s$ but possibly the last one we have $f(s) \neq 0$.

Like the algorithm that we described in the previous subsection, Algorithm 4 employs only two different variation operators, `uniform()` and `RLS(·)`, which (cf. Remark 5) are unbiased and of arity 0 and 1, respectively. It remains to show that w.h.p. Algorithm 4 queries an optimal solution after $O(n \log n)$ queries. We show correctness for the three phases. The high probability statement follows from a simple union bound over the failure probabilities.

**Phase 1: Shifting all weights to the same bin.** We query in the first step of this first phase $2n \ln n$ random bit strings $x^{(1,t_i)}$ that differ from the initial uniform solution $x^{(1,0)}$ in exactly one bit. By the coupon collector argument, with probability at least $1 - n^{-1}$, there exists for each $i \in [n]$ at least one $t_i \leq 2n \ln n$ such that $x^{(1,0)}$ and $x^{(1,t_i)}$ differ exactly in the $i$th bit. Lemma 8, which will be presented below, shows that for each string $x^{(1,\ell)} \in \{x^{(1,t)} \mid t \in [0..2n \ln n]\}$ of maximal function value $f(x^{(1,\ell)}) = \max\{f(x^{(1,t)}) \mid t \in [0..2n \ln n]\}$ it holds that $w_{\max} \in \mathcal{I}_{\max}(x^{(1,\ell)})$. In lines 6 and 7 of Algorithm 4 we fix one such $\ell$ and set $x := x^{(1,\ell)}$.

**Algorithm 4:** Unary unbiased black-box algorithm for $\textsc{Partition}_{\neq}$ with the unsigned objective function.

**Initialization:**
Sample $x^{(1,0)} \leftarrow \texttt{uniform}()$. Query $f(x^{(1,0)})$;
**Shifting all weights to one bin:**
**for** $t = 1$ **to** $2n \ln n$ **do**
  Sample $x^{(1,t)} \leftarrow \texttt{RLS}(x^{(1,0)})$ and query $f(x^{(1,t)})$;
Let $\ell \in \arg\max_{0 \le t \le 2n \ln n} f(x^{(1,t)})$;
$x \leftarrow x^{(1,\ell)}$;
**for** $t = 2n \ln n + 1$ **to** $4n \ln n$ **do**
  Sample $y \leftarrow \texttt{RLS}(x)$ and query $f(y)$;
  **if** $f(y) > f(x)$ **then** $x \leftarrow y$;
  ;
**Learning the instance** $\mathcal{I}$:
**for** $t = 1$ **to** $2n \ln n$ **do**
  Sample $x^{(2,t)} \leftarrow \texttt{RLS}(x)$ and query $f(x^{(2,t)})$;
**Optimization:**
Compute an optimal solution $(\mathcal{O}_0, \mathcal{O}_1)$ such that $w_{\max} \in \mathcal{O}_1$ offline and set $\mathcal{M} \leftarrow \mathcal{O}_1$.
**for** $t = 1$ **to** $2n \ln n$ **do**
  Sample $x^{(3,t)} \leftarrow \texttt{RLS}(x)$ and query $f(x^{(3,t)})$;
  **if** $f(x) > 2w_{\max}$ and $f(x^{(3,t)}) < f(x)$ **then**
    compute $w := (f(x) - f(x^{(3,t)}))/2$;
    **if** $w \neq w_{\max}$ and $w \in \mathcal{M}$ **then**
      $x \leftarrow x^{(3,t)}$; $\mathcal{M} \leftarrow \mathcal{M} \setminus \{w\}$;
**for** $t = 1$ **to** $n \ln n$ **do**
  Sample $x^{(4,t)} \leftarrow \texttt{RLS}(x)$ and query $f(x^{(4,t)})$;

Lemma 8 and Lemma 9 verify the following. If $y$ is created from $x$ by flipping the $i$th bit of $x$, then $f(y) > f(x)$ if and only if the $i$th heaviest weight is not in the heavier bin, i.e., $\sigma(i) \notin \mathcal{I}_{\max}(x)$.

In the second step of the first phase we aim at creating a string $x'$ with $\mathcal{I}_{\max}(x') = \mathcal{I}$. We do that by querying $y = \texttt{RLS}(x)$ and updating $x \leftarrow y$ if and only if $f(y) > f(x)$. From the statement of the previous paragraph this is the case only if the bit flip has moved the corresponding weight from the lighter to the heavier bin. Again from the coupon collector argument it follows that after an additional $2n \ln n$ iterations we have $\mathcal{I}_{\max}(x) = \mathcal{I}$, with probability at least $1 - n^{-1}$.

Hence, after a total number of $4n \ln n + 1$ iterations, we have created a bit string $x$ with $\mathcal{I}_{\max}(x) = \mathcal{I}$, with probability at least $1 - 2n^{-1}$.

**Phase 2: Learning instance $\mathcal{I}$.** As in the previous subsection we learn the different weights by performing random one-bit flips. Since the objective function reveals only unsigned function values, we need to argue how the weights of the elements that have been shifted can be derived from the function values. We distinguish two cases. Either we have $w_{\max} \geq \sum_{w \in \mathcal{I}} w/2$, in which case, by the coupon collector argument, one of the sampled strings $x^{(2,t)}$, $t \in [2n \ln n]$ is optimal (i.e., $\{w_{\max}\} = \mathcal{I}_{\max}(x^{(2,t)})$ for some $t \in [2n \ln n]$) with probability at least $1 - n^{-1}$. In this case we are immediately done. Otherwise we have that for any $t \leq 2n \ln n$ it holds that $f(x^{(2,t)}) < f(x)$ (since we are always shifting exactly one weight from the bin containing all weights to the empty one) and that the corresponding weight which has been flipped from one bin to the other is of weight $(f(x) - f(x^{(2,t)}))/2$. In this case we have, again by the coupon collector argument, that $\mathcal{I}' := \{(f(x) - f(x^{(2,t)}))/2 \mid t \in [2n \ln n]\}$ equals $\mathcal{I}$, with probability at least $1 - n^{-1}$.

**Phase 3: Creating the optimal solution.** Knowing instance $\mathcal{I}$, the algorithm computes an optimal solution $(\mathcal{O}_0, \mathcal{O}_1)$ for $\mathcal{I}$ offline, e.g., by the brute force algorithm. Note that for each $y \in \{0,1\}^n$ and its bitwise complement $\bar{y}$ it holds that $f(\bar{y}) = f(y)$. Thus, we can assume without loss of generality that $(\mathcal{O}_0, \mathcal{O}_1)$ is chosen such that $w_{\max} \in \mathcal{O}_1$.

For creating the bit string which corresponds to $(\mathcal{O}_0, \mathcal{O}_1)$, we initialize $\mathcal{M} := \mathcal{O}_1$. Throughout this phase $\mathcal{M}$ denotes the set of all weights that, in order to create the string corresponding to $(\mathcal{O}_0, \mathcal{O}_1)$, still need to be "moved" from one bin to the other. The key idea here is that we required $w_{\max} \in \mathcal{M}$ and that we do not accept the weight $w_{\max}$ to be flipped too early. This is important for the following reason.

Recall from Remark 7 that if $y$ is created from $x$ by flipping the $i$th bit in $x$ and if $f(y) < f(x)$ then the corresponding weight $\sigma^{-1}(i) \in \{((f(x) - f(y))/2), (f(x) + f(y))/2\}$. But, as long as $f(x) > 2w_{\max}$ we have $(f(x) + f(y))/2 > w_{\max}$ (unless $f(y) = 0$ in which case we are done). That is, as long as $f(x) > 2w_{\max}$ it holds in the situation above that $\sigma^{-1}(i) = (f(x) - f(y))/2$.

It is easy to verify that as soon as $f(x) \leq 2w_{\max}$ we have $\mathcal{M} = \{w_{\max}\}$. It again is the coupon collector argument which ensures with probability at least $1 - n^{-1}$ that after $2n \ln n$ iterations of the third phase we are in this situation. Thus, all we need to do now is to put $w_{\max}$ from bin $\mathcal{I}_{\max}(x)$ to the other one, i.e., we need to flip $\sigma(w_{\max})$. As for each iteration the probability to flip this position is $1/n$, we can bound the probability that we have flipped it after an additional $n \ln n$ iterations from below by $1 - (1 - 1/n)^{n \ln n} \geq 1 - 1/n$. Here we have used that for all $r \in \mathbb{R}$ we have $1 + r \leq \exp(r)$. □

Let us now prove the statements omitted in the proof of Theorem 6. We use the same notation as above.

**Remark 7.** Let $\mathcal{I}$ be an instance of PARTITION$_{\neq}$ equipped with the ordering $\sigma_{\mathcal{I}}$ and objective function $f = |f_{\mathcal{I}}|$. If $y$ has been created from $x$ by flipping the $i$th bit of $x$ and $0 \neq f(y) \neq f(x) \neq 0$, we cannot uniquely identify the corresponding weight $w_i = \sigma^{-1}(i)$. More precisely, if we do not have further knowledge on the size of the weights, there are the two possibilities

$$w_i \in \begin{cases} \{\frac{1}{2}(f(y) - f(x)), \frac{1}{2}(f(y) + f(x))\}, & \text{if } f(y) > f(x), \\ \{\frac{1}{2}(f(x) - f(y)), \frac{1}{2}(f(y) + f(x))\}, & \text{if } f(y) < f(x). \end{cases}$$

**Proof.** The first statement clearly follows from the second. We give an example nevertheless, to illustrate the situation. Let $\mathcal{I} := \{1, 2, 3, 4, 6\}$, let $\sigma_{\mathcal{I}}$ be the ordering of $\mathcal{I}$, and let $x := (1, 0, 0, 0, 1)$, i.e., weights 1 and 6 are in one bin and the other weights are in the second bin. Then $f(x) = |7 - 9| = 2$. Now both bit strings $y := (0, 0, 0, 0, 1)$ and $z := (1, 0, 1, 0, 1)$ have Hamming distance 1 from $x$ and both have function value $f(y) = |6 - 10| = 4 = f(z)$. Hence, knowing $x$, knowing that $|x - z|_1 = 1$, and knowing the function values $f(x)$ and $f(z)$ does not suffice to compute the bit in which $x$ and $z$ differ.

To prove the second statement we assume first that $f(y) > f(x)$. If weight $w_i \notin \mathcal{I}_{\max}(x)$ then clearly we have $f(y) = f(x) + 2w_i$. We therefore assume that $w_i \in \mathcal{I}_{\max}(x)$. Then $w_i \in \mathcal{I}_{\max}(y)$ for otherwise $f(y) = f(x) - 2w_i < f(x)$ contradicting our assumption $f(y) > f(x)$. Hence, $f(y) = 2w_i - f(x)$. Note that in this case we necessarily have $w_i > f(x)$.

Assume now that $f(y) < f(x)$. In this case we must have $w_i \in \mathcal{I}_{\max}(x)$ for otherwise $f(y) = f(x) + 2w_i > f(x)$. If $w_i \notin I_{\max}(y)$ then clearly we have $f(y) = f(x) - 2w_i$. On the other hand, $w_i \in I_{\max}(y)$ implies $w_i > f(x)/2$ and $f(y) = 2w_i - f(x)$.

This enumerates all possible combinations and the claim follows. □

**Lemma 8.** *Let $\mathcal{I}$ be an instance of* PARTITION$_{\neq}$*, let $\sigma = \sigma_{\mathcal{I}}$ be its ordering, and let $f = |f_{\mathcal{I}}|$. Furthermore, let $x^{(0)} \in \{0,1\}^n$ and for each $i \in [n]$ let $x^{(i)}$ be created from $x^{(0)}$ by flipping the $i$th bit.*

*If we choose $\ell \in [0..n]$ such that $f(x^{(\ell)}) = \max\{f(x^{(t)}) \mid t \in [0..n]\}$, then $w_{\max} \in \mathcal{I}_{\max}(x^{(\ell)})$.*

**Proof.** We assume that $w_{\max} \notin \mathcal{I}_{\max}(x^{(\ell)})$ to show the contrapositive. If $\ell = 0$, we can flip the bit corresponding to $w_{\max}$ (by our assumption on $\sigma$ this is the $n$th bit) in $x^{(0)}$ to get $f(x^{(n)}) = f(x^{(0)}) + 2w_{\max} > f(x^{(0)})$. Similarly, if $\ell = n$ then $f(x^{(0)}) = f(x^{(n)}) + 2w_{\max}$. All other values of $\ell$ imply $w_{\max} \notin \mathcal{I}_{\max}(x^{(0)})$ (by the assumption that $w_{\max} \notin \mathcal{I}_{\max}(x^{(\ell)})$) and thus, $f(x^{(n)}) - f(x^{(0)}) = 2w_{\max}$. But since the weights are pairwise different, $\sigma^{-1}(\ell) < w_{\max}$ and thus $f(x^{(\ell)}) - f(x^{(0)}) < 2w_{\max}$. □

The previous lemma has shown that the largest weight $w_{\max}$ is in the larger of the two bins of $x^{(\ell)}$. The following lemma shows that if we have iteratively increased the value of $x^{(\ell)}$ through 1-bit flips, we only have shifted weights from the smaller bin to the larger one.

**Lemma 9.** *Let $\mathcal{I}$ be an instance of* PARTITION$_{\neq}$*, let $\sigma := \sigma_{\mathcal{I}}$ be the ordering of $\mathcal{I}$, and $f := |f_{\mathcal{I}}|$.*

(i) *If $x \in \{0,1\}^n$ with $f(x) \geq w_{\max}$, then for all $i \in [n]$ we have $f(x \oplus e_i) > f(x)$ if and only if $w_i := \sigma^{-1}(i) \notin \mathcal{I}_{\max}(x^{\ell})$.*

(ii) *For $x^{(0)}, \ldots, x^{(n)}$ and $\ell$ as in Lemma 8 we have $f(x^{(\ell)}) \geq w_{\max}$.*

**Proof.** (i). Let $x \in \{0,1\}^n$ with $f(x) \geq w_{\max}$ and let $i \in [n]$. Clearly, if $w_i \notin \mathcal{I}_{\max}(x)$ then $f(x \oplus e_i) = f(x) + 2w_i > f(x)$. On the other hand, if $w_i \in \mathcal{I}_{\max}(x)$ then either $f(x \oplus e_i) = f(x) - 2w_i < f(x)$ or $f(x \oplus e_i) = 2w_i - f(x) \leq 2w_i - w_{\max} \leq w_{\max} \leq f(x)$.

(ii). If $w_{\max} \notin \mathcal{I}_{\max}(x^{(0)})$, then $\ell = n$ since for all $i \in [n]$ we have

$$f\left(x^{(n)}\right) = f\left(x^{(0)}\right) + 2w_{\max} \geq f\left(x^{(0)}\right) + 2w_i \geq f\left(x^{(i)}\right).$$

The above calculation immediately yields $f(x^{(\ell)}) > w_{\max}$.

Therefore, we may thus assume that $w_{\max} \in \mathcal{I}_{\max}(x^{(0)})$. To show the contrapositive, let us also assume that $f(x^{(\ell)}) < w_{\max}$. Then $f(x^{(0)}) \leq f(x^{(\ell)}) < w_{\max}$ and thus $f(x^{(n)}) = 2w_{\max} - f(x^{(0)}) > w_{\max} > f(x^{(\ell)})$, contradicting the choice of $\ell$. Thus, $f(x^{(\ell)}) \geq w_{\max}$. □

It is not difficult to see that already with 3-ary variation operations it is possible to access every bit position in a linear number of iterations. Hence, a small modification of Algorithm 3 solves PARTITION$_{\neq}$ and even PARTITION in a linear number of steps, using only unbiased variation operators of arity at most 3. This implies, in particular, that the unrestricted black-box complexity of PARTITION is linear in $|\mathcal{I}|$. The latter can be seen alternatively by the fact that we can learn the weights by querying first the all-zeros bit string and then the $n$ different unit vectors $(0, \ldots, 0, 1, 0, \ldots, 0)$.

**Remark 10.** The 3-ary unbiased black-box complexity of PARTITION is at most linear in the size $|\mathcal{I}|$ of the input set $\mathcal{I}$.

## 4. Conclusions

We have shown that the unbiased black-box model allows for algorithms which solve the optimization version of the $\mathcal{NP}$-hard partition problem in a polynomial number of queries, even if the arity of the algorithms is restricted to one. Our result indicates that the unbiased black-box model, while clearly closer to the truth than the unrestricted one, still does not provide a complete picture on how difficult it is to solve a given problem via randomized search heuristics. It seems that further restrictions to the power of the algorithms are needed to obtain meaningful results. A recent step into this direction is the work by the first two authors [14], who, following a suggestion by Nikolaus Hansen, investigate a black-box model where the algorithm can only compare the quality of solutions, but has no access to the absolute function values. We do not know yet the black-box complexity of, e.g., Partition in this new model.

We should note, though, that observing smaller-than-expected black-box complexities does not always reveal a weakness of the black-box complexity model regarded. In [7], the authors exhibit the first $O(n \log n)$ crossover-based (unbiased) evolutionary algorithm for the OneMax function class, which shows that the $O(n \log n)$ bound for the 2-ary unbiased black-box complexity of OneMax found in [8] is not as unnatural as it might have seemed at first.

## Acknowledgements

This work was done while Benjamin Doerr and Timo Kötzing were with the Max Planck Institute for Informatics (MPII), Saarbrücken, Germany and Carola Doerr was with the MPII and the LIAFA, Université Paris Diderot (Paris 7), France.

Carola Doerr gratefully acknowledges support from a Google Europe Fellowship in Randomized Algorithms, from the Alexander von Humboldt Foundation, and the Agence Nationale de la Recherche (project ANR-09-JCJC-0067-01).

Timo Kötzing was supported by the Deutsche Forschungsgemeinschaft (DFG) under grant NE 1182/5-1.

Parts of this work have been done during the Dagstuhl seminar 10361 “Theory of Evolutionary Algorithms”.

## References

[1] Peyman Afshani, Manindra Agrawal, Benjamin Doerr, Carola Doerr, Kasper Green Larsen, Kurt Mehlhorn, The query complexity of finding a hidden permutation, in: Space-Efficient Data Structures, Streams, and Algorithms—Papers in Honor of J. Ian Munro on the Occasion of His 66th Birthday, in: Lecture Notes in Computer Science, vol. 8066, Springer, 2013, pp. 1–11.
[2] David Aldous, Minimization algorithms and random walk on the $d$-cube, Ann. Probab. 11 (1983) 403–413.
[3] Dana Angluin, Queries and concept learning, Mach. Learn. 2 (1988) 319–342.
[4] Anne Auger, Benjamin Doerr, Theory of Randomized Search Heuristics, World Scientific, 2011.
[5] Nader H. Bshouty, Richard Cleve, Ricard Gavaldà, Sampath Kannan, Christino Tamon, Oracles and queries that are sufficient for exact learning, J. Comput. Syst. Sci. 52 (1996) 421–433.
[6] Edward G. Coffman, Ward Whitt, Recent asymptotic results in the probabilistic analysis of schedule makespans, in: Philippe Chrétienne, Edward G. Coffman, Jan Karel Lenstra, Zhen Liu (Eds.), Scheduling Theory and Its Applications, Wiley, 1995, pp. 15–31.
[7] Benjamin Doerr, Carola Doerr, Franziska Ebel, Lessons from the black-box: fast crossover-based genetic algorithms, in: Proc. of the Annual Genetic and Evolutionary Computation Conference, GECCO'13, ACM, 2013, pp. 781–788.
[8] Benjamin Doerr, Daniel Johannsen, Timo Kötzing, Per Kristian Lehre, Markus Wagner, Carola Winzen, Faster black-box algorithms through higher arity operators, in: Proc. of the 11th ACM Workshop on Foundations of Genetic Algorithms, FOGA'11, ACM, 2011, pp. 163–172.
[9] Stefan Droste, Thomas Jansen, Karsten Tinnefeld, Ingo Wegener, A new framework for the valuation of algorithms for black-box optimization, in: Proc. of the 7th Workshop on Foundations of Genetic Algorithms, FOGA'03, Morgan Kaufmann, 2003, pp. 253–270.
[10] Stefan Droste, Thomas Jansen, Ingo Wegener, Upper and lower bounds for randomized search heuristics in black-box optimization, Theory Comput. Syst. 39 (2006) 525–544.
[11] Benjamin Doerr, Timo Kötzing, Johannes Lengler, Carola Winzen, Black-box complexities of combinatorial problems, Theor. Comput. Sci. 471 (2013) 84–106.
[12] Benjamin Doerr, Timo Kötzing, Carola Winzen, Too fast unbiased black-box algorithms, in: Proc. of the 13th Annual Genetic and Evolutionary Computation Conference, GECCO'11, ACM, 2011, pp. 2043–2050.
[13] Benjamin Doerr, Carola Winzen, Black-box complexity: breaking the $O(n \log n)$ barrier of LeadingOnes, in: Artificial Evolution (EA'11), Revised Selected Papers, in: Lecture Notes in Computer Science, vol. 7401, Springer, 2012, pp. 205–216.
[14] Benjamin Doerr, Carola Winzen, Ranking-based black-box complexity, Algorithmica 68 (2014) 571–609.
[15] Benjamin Doerr, Carola Winzen, Reducing the arity in unbiased black-box complexity, Theor. Comput. Sci. 545 (2014) 108–121, http://dx.doi.org/10.1016/j.tcs.2013.05.004, forthcoming.
[16] Paul Erdős, Alfréd Rényi, On two problems of information theory, Magy. Tud. Akad. Mat. Kut. Intéz. Közl. 8 (1963) 229–243.
[17] Johannes B.G. Frenk, Alexander H.G. Rinnooy Kan, The rate of convergence to optimality of the LPT rule, Discrete Appl. Math. 14 (1986) 187–197.
[18] Michael R. Garey, David S. Johnson, Computers and Intractability, A Guide to the Theory of NP-Completeness, W. H. Freeman & Co., 1990.
[19] Dirk Hausmann, Bernhard Korte, Lower bounds on the worst-case complexity of some oracle algorithms, Discrete Math. 24 (1978) 261–276.
[20] Juraj Hromkovič, Algorithmics for Hard Problems: Introduction to Combinatorial Optimization, Randomization, Approximation, and Heuristics, Springer, 2001.
[21] Thomas Jansen, Analyzing Evolutionary Algorithms—The Computer Science Perspective, Springer, 2013.
[22] Richard M. Karp, Reducibility among combinatorial problems, in: Proc. of a Symposium on the Complexity of Computer Computations, Plenum Press, 1972, pp. 85–103.
[23] Donna Crystal Llewellyn, Craig Tovey, Michael Trick, Local optimization on graphs, Discrete Appl. Math. 23 (1989) 157–178.
[24] Per Kristian Lehre, Carsten Witt, Black-box search by unbiased variation, Algorithmica 64 (2012) 623–642.
[25] Erich Novak, Ian H. Sloan, Joseph F. Traub, Henryk Woźniakowski, Essays on the Complexity of Continuous Problems, European Mathematical Society (EMS), Zürich, 2009.

[26] Frank Neumann, Carsten Witt, Bioinspired Computation in Combinatorial Optimization – Algorithms and Their Computational Complexity, Springer, 2010.
[27] Jonathan Rowe, Michael Vose, Unbiased black box search algorithms, in: Proc. of the 13th Annual Genetic and Evolutionary Computation Conference, GECCO'11, ACM, 2011, pp. 2035–2042.
[28] Carsten Witt, Worst-case and average-case approximations by simple randomized search heuristics, in: Proc. of the 22nd Annual Symposium on Theoretical Aspects of Computer Science, STACS'05, Springer, 2005, pp. 44–56.