Too Fast Unbiased Black-Box Algorithms

Benjamin Doerr
Max-Planck-Institut für Informatik
Saarbrücken, Germany

Timo Kötzing
Max-Planck-Institut für Informatik
Saarbrücken, Germany

Carola Winzen
Max-Planck-Institut für Informatik
Saarbrücken, Germany

ABSTRACT

Unbiased black-box complexity was recently introduced as a refined complexity model for randomized search heuristics (Lehre and Witt, GECCO 2010). For several problems, this notion avoids the unrealistically low complexity results given by the classical model of Droste, Jansen, and Wegener (Theor. Comput. Sci. 2006).

In this work, we show that for two natural problems the unbiased black-box complexity remains artificially small. For the classical $\textsc{Jump}_k$ test function class and for a subclass of the well-known Partition problem, we give mutation-only unbiased black-box algorithms having complexity $O(n \log n)$. Since the first problem usually needs $\Theta(n^k)$ function evaluations to be optimized by standard heuristics and the second is even $\mathcal{NP}$-complete, these black-box complexities seem not to indicate the true difficulty of the two problems for randomized search heuristics.

Categories and Subject Descriptors

F.2 [**Theory of Computation**]: Analysis of Algorithms and Problem Complexity

General Terms

Theory, algorithms

Keywords

Black-box complexity, running time analysis, theory

1. INTRODUCTION

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

GECCO'11, July 12–16, 2011, Dublin, Ireland.

For evolutionary algorithms, or more broadly, randomized search heuristics, the classical complexity notions are not very useful. They assume that the algorithm is fully aware of the complete problem instance. This is very different from the typical application of a randomized search heuristic. Here, we usually do not have the full problem data available, either because we really do not have it, or because we do not want to fully exploit it (e.g., due to its size or complexity). Rather, the typical randomized search heuristic only obtains information about the problem by learning the fitness (quality) of the search points it generates.

The notion of black-box complexity tries to build a complexity theory for such settings. In simple words, the black-box complexity of a problem is the number of fitness evaluations necessary to solve the problem. This notion was introduced in [8] (see [9] for the journal version). Earlier related work exists, e.g., [1, 12, 16].

As observed already in [9], this unrestricted notion of black-box complexity, by allowing too powerful algorithms, yields unrealistically low complexities. Examples include a black-box complexity of less than 2 for the (not permutation invariant) BinaryValue function class or a polynomial black-box complexity for the $\mathcal{NP}$-complete Max-Clique problem.

After these results, the quest for a complexity theory for randomized search heuristics seemed to have come to an early end (apart from the again unrealistically low $\Theta(n/\log n)$ bound for the OneMax function class due to Anil and Wiegand [2]).

Black-box complexity was revived by Lehre and Witt in their best-paper awarded GECCO 2010 paper [15]. To overcome the drawbacks of the previous unrestricted black-box model, they restricted the class of admitted black-box optimization algorithms in a natural way, still admitting a large class of classically used algorithms. In their unbiased black-box complexity model, they require that all solution candidates must be obtained by variation operators. In addition, 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). This model, in addition to excluding some highly artificial algorithms, also admits a notion of arity. A k-ary unbiased black-box algorithm is one that employs only such variation operators that take up to k arguments. This allows, e.g., to talk about mutation-only algorithms (arity one).

For several function classes, the unbiased 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. The mutation only black-box complexity of any class of functions having a unique global optimum is $\Omega(n \log n)$ [15]. The (permutation invariant) LeadingOnes function class, having an unrestricted black-box complexity of $O(n \log n)$ proven in [8], now has a mutation-only black-box complexity of $\Theta(n^2)$ [15], matching the run time (defined as expected number of function evaluations until an optimal search point is queried) of standard randomized search heuristics. Hence, for mutation-only black-box algorithms, the unbiased model leads to much better complexity estimates.

When higher arity-variation operators are used, smaller, but still not completely unrealistic complexities were observed in [5]. Among others, an $O(n)$ binary (that is, allowing crossover-type variation operators) complexity for the OneMax class and an $O(n \log n)$ binary complexity for the LeadingOnes class were shown.

In this paper, we show that also the mutation-only unbiased black-box complexity can be ridiculously small. For the $\textsc{Jump}_k$ function class, typically needing $\Omega(n^k)$ function evaluations to be solved via a randomized search heuristic, we present a mutation-only unbiased black-box algorithm solving the problem in $O(n \log n)$ queries. Similarly, for an $\mathcal{NP}$-hard subclass of the Partition problem, we again prove a unary black-box complexity of $O(n \log n)$.

These results indicate that, in addition to the progress made in [15], more effort is needed to define a complexity model that gives realistic complexity statements for a broader class of problems.

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.

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 bitwise complement of x (i.e., for all $i \in [n]$ we have $\bar{x}_i = 1 - x_i$). The bitwise exclusive-OR is denoted by $\oplus$. When we say that y is created from x by flipping the i-th bit in x, we require $y = x \oplus e_i$ where e_i denotes the i-th unit vector.

For any set S we denote by 2^S the power set of S, i.e., the set of all subsets of S.

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 log 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 [11, 13]). 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 to be solved by learning the objective value of possible solutions ("search points"). The objective function is given by an oracle or as a *black-box*. Using this oracle, the algorithm may query the objective value of all possible solutions, but any such query does only return this search point's objective value and no other information about the objective function.

For simplicity, we shall restrict ourselves to real-valued objective functions defined on the set $\{0,1\}^n$ of bit strings of length n (so called *pseudo-Boolean functions*). This is motivated by the fact that many evolutionary algorithms use such a representation. For results on more general search spaces confer [6].

Naturally, we do allow that the algorithms use random decisions. From the black-box concept, it follows 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 objective value ("fitness") from the oracle. This leads to the scheme of Algorithm 1, which we call an *unrestricted black-box algorithm*.

As performance measure of a black-box algorithm 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 fitness of a search point is more costly than the generation of a new search point. Since we mainly talk about randomized algorithms, we regard 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 introduced by Droste, Jansen, and Wegener [9].

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

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 of f as first action shows this bound.

This drawback of the unrestricted black-box model inspired Lehre and Witt [15] 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 the model allows for most of the commonly studied search heuristics, such as many $(\mu+\lambda)$ and (μ,λ) evolutionary algorithms (EAs), simulated annealing algorithms, the Metropolis algorithm, and the Randomized Local Search algorithm.

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

$(i)\ \forall x, z \in \{0,1\}^n$:

$$D(x \mid y^{(1)}, \dots, y^{(k)}) = D(x \oplus z \mid y^{(1)} \oplus z, \dots, y^{(k)} \oplus z)\,;$$

$(ii)\ \forall x \in \{0,1\}^n\ \forall \sigma \in S_n$:

$$D(x \mid y^{(1)}, \dots, y^{(k)}) = D(\sigma(x) \mid \sigma(y^{(1)}), \dots, \sigma(y^{(k)}))\,.$$

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 one. 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.

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

Note that, for all $k \le \ell$, each k-ary unbiased black-box algorithm is contained in the ℓ-ary unbiased black-box model.

As mentioned in the introduction, Lehre and Witt [15] 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 on. Recall that, as pointed out above, the unrestricted black-box complexity of any such function is 1. For results on higher arity models refer to the work of Doerr et al. [5].

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 number s 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 after 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 call the s iterations of A a *run* of A. 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 $\Pr[X_i = 1] \ge 1 - c$. 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. □

3. JUMP FUNCTIONS

In this section we analyze the unbiased black-box complexity of the so-called *jump* functions: for all $k < n/2$, let $\textsc{Jump}_k$ denote the fitness function defined by

$$\textsc{Jump}_k(x) = \begin{cases} n, & \text{if } |x|_1 = n; \\ |x|_1, & \text{if } k < |x|_1 < n-k; \\ 0, & \text{otherwise,} \end{cases}$$

for all $x \in \{0,1\}^n$.

We show that for every k, k constant, the black-box complexity of $\textsc{Jump}_k$ is surprisingly low. Key to the analysis is the following lemma. It shows that one can compute, with high probability, the OneMax value of any search point x with few black-box calls to $\textsc{Jump}_k$. With this, we can orient ourselves on the large plateau surrounding the optimum and thus revert to the problem of optimizing OneMax.

We collect these computations in a subroutine, to be called by black-box algorithms.

LEMMA 3. *For all constants k and c, there is a unary unbiased subroutine s using $c+1$ queries to $\textsc{Jump}_k$ such that, for all bit strings x, $s(x) = \textsc{OneMax}(x)$ with probability $1 - O(n^{-c})$.*

PROOF. We use a unary unbiased variation operator $\texttt{flip}_k$, which samples uniformly a k-neighbor (a bit string which differs in exactly k positions) of the argument. Next we give the subroutine s, which uses $\textsc{Jump}_k$ to approximate OneMax as desired, see Algorithm 3. Intuitively, the subroutine samples c bit strings in the k-neighborhood of x; if $|x|_1 \ge n - k$ then it is likely that at least once only 1s of x have been flipped, leading to a $\textsc{Jump}_k$-value of $|x|_1 - k$; as no sample will have a lower $\textsc{Jump}_k$-value, adding k to the minimum non-0 fitness of one of the sampled bit strings gives the desired output. The case of x with $|x|_1 \le k$ is analogous.

Clearly, the subroutine is correct with certainty on all x with $k < |x|_1 < n - k$. The other two cases are symmetric, so let x with $|x|_1 \ge n - k$ be given. Obviously, the return value of the subroutine is correct if and only if at least one of the c samples flips only 1s in x. We denote the probability

Algorithm 3: Simulation of OneMax using the jump function.

subroutine $s(x)$ **is**
  if $\textsc{Jump}_k(x) \neq 0$ **then output** $\textsc{Jump}_k(x)$;
  $M \leftarrow \{\textsc{Jump}_k(\texttt{flip}_k(x)) \mid i \in [c]\}$;
  if $\max(M) < n/2$ **then** $m \leftarrow \max(M) - k$;
  else $m \leftarrow \min(M \setminus \{0\}) + k$;
  output m;

of this event with p. We start by bounding the probability that a single sample flips only 1s. We choose which k bits to flip iteratively so that, after i iterations, there are at least $n-k-i$ bit positions with a 1 out of $n-i$ unchosen bit positions left to choose. This gives the bound of

$$\left(\tfrac{n-k}{n}\right) \cdot \left(\tfrac{n-k-1}{n-1}\right) \cdots \left(\tfrac{n-k-(k-1)}{n-(k-1)}\right) = \prod_{i=0}^{k-1} \left(1 - \tfrac{k}{n-i}\right) \geq \left(1 - \tfrac{k}{n-k}\right)^k \geq \left(1 - \tfrac{k^2}{n-k}\right),$$

using Bernoulli's inequality. Thus, we have

$$p \geq 1 - \left(\frac{k^2}{n-k}\right)^c.$$

□

Now we can use the subroutine from Lemma 3 to turn results on the unbiased black-box complexity for OneMax into results on $\textsc{Jump}_k$ for constant k.

Theorem 4. *For constant k, the unbiased black-box complexity of $\textsc{Jump}_k$ is*

- $O(n \log n)$*, for unary variation operators;*
- $O(n/\log m)$*, for m-ary variation operators with $2 \leq m \leq n$;*
- $O(n/\log n)$*, for $*$-ary variation operators.*

Proof. Note that the above black-box complexities claimed for $\textsc{Jump}_k$ are shown for OneMax in [5]. We use Lemma 3 with $c = 4$ and run the unbiased black-box algorithms of the appropriate arity for OneMax; all sampled bit strings are evaluated using the subroutine s. Thus, this algorithm samples as if working on OneMax and finds the bit string with all 1s after the desired number of iterations. Note that, for up to $n \log n$ uses of s, we expect no more than $n \log n O(n^{-4}) \leq O(n^{-2})$ incorrect evaluations of s. Therefore, there is a small chance of failing, and the claim follows from Remark 2. □

Note that the subroutine from Lemma 3 requires to know the parameter k; however, this subroutine can be modified to work without that knowledge as follows. The first time that the subroutine samples a search point with fitness 0 it will determine k; after knowing k, it will work as before (and before sampling a search point of fitness 0, it does not need to know). The parameter k is determined by sampling sufficiently many i-neighbors of the search point with fitness 0, starting with $i = 1$ and stopping when a search point with fitness $\neq 0$ is found. This search point will have maximum fitness among all non-optimal search points, equal to $n-k-1$. From this fitness and n, the subroutine can infer k.

There may be cases when one of the algorithms implicit in the proof of Theorem 4 never samples a search point with fitness 0 and does not have to determine k. In this case, such an algorithm will optimize the target function without completely learning it. However, in a second phase after finding the optimum, an algorithm could determine k with a binary search, as k equals the largest distance from the optimum at which all and any search point has fitness 0. This phase requires $O(\log(n))$ queries.

4. PARTITION

One criticism received by the unrestricted previous black-box model is that there are $\mathcal{NP}$-hard problems which have a polynomial black-box complexity. As an example let us consider the well-known Clique problem. Although it is known to be $\mathcal{NP}$-hard, Droste et al. [9] show that Max-Clique has an unrestricted black-box complexity of at most $\binom{n}{2}+1 = \Theta(n^2)$. That is, even the optimization problem can be solved by an unrestricted black-box algorithm in $\Theta(n^2)$ queries.

In this section we prove that in the unbiased black-box model, too, there are $\mathcal{NP}$-hard problems having a polynomial black-box complexity.

To this end we consider an $\mathcal{NP}$-hard subclass of the Partition problem. Whereas the decision version of the Partition problem asks the question "Given a multiset $\mathcal{I}$ of positive integers ("weights"), is it 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 optimization version asks for 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.

It is known that Partition permits heuristics which solve many instances of the problem in a polynomial number of fitness evaluations. For example, Frenk and Kan [10] 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. For example, Witt [17] 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 can be found in [4] and [17].

The decision version of Partition has been shown to be $\mathcal{NP}$-hard, cf. [11, 14]. It actually is one of the most famous $\mathcal{NP}$-hard problems. Thus, assuming $\mathcal{P} \neq \mathcal{NP}$, the decision problem does not allow a polynomial-time algorithm. Note that this also implies that the optimization problem cannot be solved in polynomial time.

It is easily seen that Partition remains $\mathcal{NP}$-hard if we restrict the problem to instances with all weights distinct.

Lemma 5 (folklore). Partition *remains $\mathcal{NP}$-hard when restricted to instances $\mathcal{I}$ with $v \neq w$ for all $v, w \in \mathcal{I}$.*

Proof. Omitted for space restrictions. □

In the following, let $\textsc{Partition}_{\neq}$ be the subclass of Partition instances with pairwise different weights. There is no one best way of how to consider $\textsc{Partition}_{\neq}$ as an optimization problem. In the following we present two different fitness function and show that for both problems a polynomial unary unbiased black-box complexity can be achieved. In Section 4.1 we consider a signed fitness function where we learn from the fitness values which bin is the heavier one.

We show that the unary unbiased black-box complexity of PARTITION$_{\neq}$ under this fitness function is $O(n \log n)$.

In Section 4.2 we then consider an unsigned fitness function. A priori we do have less information than in the situation of Section 4.1. However, we are still able to prove the same asymptotic bound. This second fitness function is probably the more natural one but note that the key arguments for our upper bound are essentially the same.

4.1 The Signed Fitness Function

As mentioned above, we consider in this section a signed fitness function for PARTITION$_{\neq}$. Given an instance $\mathcal{I}$ of PARTITION$_{\neq}$, we set $\mathcal{F}_{\mathcal{I}} := \{(\mathcal{I}_0, \mathcal{I}_1) \in 2^{\mathcal{I}} \times 2^{\mathcal{I}} \mid \mathcal{I}_0 \dot\cup \mathcal{I}_1 = \mathcal{I}\}$, the set of all feasible solutions of $\mathcal{I}$. We define the (signed) fitness function to measures the quality of the queried solutions via

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

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

Since unbiased black-box complexity typically requires the search space $\{0,1\}^n$, let us fix some enumeration $\sigma : \mathcal{I} \to [n]$ of the elements of $\mathcal{I}$. To ease readability, let σ be the ordering of the elements in $\mathcal{I}$, i.e., $\sigma(v) < \sigma(w)$ for all $v, w \in \mathcal{I}$ with $v < w$. 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 $\mathcal{F}_{\mathcal{I}}$. Therefore, we let

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

THEOREM 6. *The unary unbiased black-box complexity of* PARTITION$_{\neq}$ *modeled via the signed fitness functions $f_{\mathcal{I}}$ is $O(n \log n)$, where $n := |\mathcal{I}|$ denotes the size of the input set $\mathcal{I}$.*

In the proof of Theorem 6 we will apply only two variation operators, namely `uniform`(), which samples a bit string $x \in \{0,1\}^n$ uniformly at random and `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. Note that `RLS`$(\cdot)$ equals `flip`$_1(\cdot)$ as defined in Section 3. The following is straightforward to verify from the definition of unbiased variation operators.

REMARK 7. `uniform`() *is a (0-ary) unbiased variation operator.* `RLS`$(\cdot)$ *is a unary unbiased variation operator.*

PROOF OF THEOREM 6. We need to show that there exists an algorithm which, for any instance $\mathcal{I}$ of PARTITION$_{\neq}$, needs an expected $O(|\mathcal{I}| \log |\mathcal{I}|)$ iterations 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. We shall show that Algorithm 4 satisfies this. As mentioned above, it only employs two different variation operators, `uniform`() and `RLS`$(\cdot)$, both unbiased and of arity at most 1.

Let us fix some instance $\mathcal{I}$ of PARTITION$_{\neq}$ and let $|\mathcal{I}| =: n$. Abbreviate $f := f_{\mathcal{I}}$.

Let us now comment on the different steps of the algorithm.

After an expected number of $(1 + o(1))n \log n$ 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 [3]) the expected number of queries until we have flipped each bit position of $x^{(0)}$ at least once is $(1 + o(1))n \log n$. Thus, we can expect that we need $t^* = (1 + o(1))n \log n$ queries until $\mathcal{W}_{t^*} = \mathcal{I}$. That is, we can assume to have learned all n different weights in $\mathcal{I}$ in $(1 + o(1))n \log n$ queries.

Algorithm 4: Unary unbiased black-box algorithm for PARTITION$_{\neq}$ with the signed fitness function

```
Initialization:
Sample x^(0) ← uniform(). Query f(x^(0));
Initialize t ← 0 and , I'_0, I'_1, W_0 = ∅;
Learning the integers:
while |W_t| < n do
    t ← t + 1;
    Sample x^(t) ← RLS(x^(0)). Query f(x^(t));
    Update W_t ← W_{t−1} ∪ {|f(x^(0)) − f(x^(t))|/2};
    if f(x^(0)) > f(x^(s)) then
        I'_0 ← I'_0 ∪ {|f(x^(0)) − f(x^(s))|/2};
    else I'_1 ← I'_1 ∪ {|f(x^(0)) − f(x^(s))|/2};
Optimization:
Offline compute an optimal solution (O_0, O_1) and set
M ← {w ∈ O_0 | w ∉ I'_0} ∪ {w ∈ O_1 | w ∉ I'_1}, the set of
integers that need to be moved;
Set z ← x^(0);
while |M| > 0 do
    Sample y ← RLS(z). Query f(y);
    if w := |f(y) − f(z)|/2 ∈ M then
    z ← y and M ← M\{w};
```

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 $\mathcal{M} \leftarrow \mathcal{M} \setminus \{w\}$.

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

Putting everything together, we see that, despite using only unary unbiased variation operators, we can opti-

mize any instance $\mathcal{I}$ of the PARTITION$_{\neq}$ problem modeled via the signed fitness function in an expected number of $2(1+o(1))n\log n = O(n\log n)$ queries. □

4.2 The Unsigned Fitness Function

One might dislike the fact that in the proof of Theorem 6 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 6 if we only allow the latter, unsigned fitness 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 fitness values $|f_{\mathcal{I}}(x)|$ and $|f_{\mathcal{I}}(y)|$ the weight of the flipped bit, cf. Remark 9. This results in a more complex procedure to learn all the weights.

THEOREM 8. *The unary unbiased black-box complexity* PARTITION$_{\neq}$ *with respect to* $|f_{\mathcal{I}}|$ *is* $O(n\log n)$. *Here,* $n := |\mathcal{I}|$ *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 8 to Lemmas, which will be presented after the proof of the main Theorem.

PROOF OF THEOREM 8. By Remark 2 it suffices to show that there exists an algorithm that, for any instance $\mathcal{I}$ of PARTITION$_{\neq}$, with high probability (w.h.p.), needs only $O(|\mathcal{I}|\log|\mathcal{I}|)$ fitness queries until it queries an optimal search point.

Let us fix an instance $\mathcal{I}$ of PARTITION$_{\neq}$, let $n := |\mathcal{I}|$ and abbreviate $f := |f_{\mathcal{I}}|$, where $f_{\mathcal{I}}$ is defined as in Section 4.1.

For readability purposes let us introduce the following notation. Note, however, that these values are a priori not identifiable for the algorithm. First note that, using the notation from Section 4.1, each $x \in \{0,1\}^n$ corresponds to a solution $(\mathcal{I}_0(x), \mathcal{I}_1(x)) \in \mathcal{F}_{\mathcal{I}}$. 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 sum of the weights and let $\mathcal{I}_{\max}(x)$ be the set of weights belonging to the bin of larger weight, that is $\mathcal{I}_{\max} := \mathcal{I}_0$ if $\mathcal{S}_0(x) \geq \mathcal{S}_1(x)$ and $\mathcal{I}_{\max}(x) = \mathcal{I}_1$ otherwise. We call $\mathcal{I}_{\max}$ the "heavier" bin and we call the other one the "lighter" bin.

Lastly, let $w_{\max} = \max \mathcal{I}$ be the maximum weight of instance $\mathcal{I}$.

The general approach of Algorithm 5 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 $4n\log n$ queries. Next, we perform $2n\log n$ 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 $3n\log n$ iterations w.h.p.

If in any iteration of Algorithm 5 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 the last one we have $f(s) \neq 0$.

Algorithm 5 employs only two different variation operators, `uniform`() and `RLS`$(\cdot)$, which (cf. Remark 7) are unbiased and of arity at most 1. It remains to show that

Algorithm 5: Unary unbiased black-box algorithm for PARTITION$_{\neq}$ with the unsigned fitness function

Initialization:
Sample $x^{(1,0)} \leftarrow$ `uniform`(). Query $f(x^{(1,0)})$;
Shifting all weights to one bin:
for $t = 1$ **to** $2n\log n$ **do**
  Sample $x^{(1,t)} \leftarrow$ `RLS`$(x^{(1,0)})$ and query $f(x^{(1,t)})$;
Let $\ell \in \arg\max_{0\leq t\leq 2nlogn} f(x^{(1,t)})$
$x \leftarrow x^{(1,\ell)}$;
for $t = 2n\log n + 1$ **to** $4n\log n$ **do**
  Sample $y \leftarrow$ `RLS`(x) and query $f(y)$;
  if $f(y) > f(x)$ **then** $x \leftarrow y$;
Learning the instance $\mathcal{I}$:
for $t = 1$ **to** $2n\log n$ **do**
  Sample $x^{(2,t)} \leftarrow$ `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** $2n\log n$ **do**
  Sample $x^{(3,t)} \leftarrow$ `RLS`(x) and query $f(x^{(3,t)})$;
  if $f(x) > 2w_{\max}$ and $f(x^{(3,t)}) < f(x)$ **then**
    compute $w := \big(f(x) - f(x^{(3,t)})\big)/2$;
    if $w \neq w_{\max}$ and $w \in \mathcal{M}$ **then**
      $x \leftarrow x^{(3,t)}$; $\mathcal{M} \leftarrow \mathcal{M}\backslash\{w\}$;
for $t = 1$ **to** $n\log n$ **do**
  Sample $x^{(4,t)} \leftarrow$ `RLS`(x) and query $f(x^{(4,t)})$;

w.h.p. Algorithm 5 queries an optimal solution after at most $O(n\log n)$ queries. We show correctness for the three phases. The high probability statement follows from a simple union bound over the failure probabilities.

Shifting all weights to one bin. It follows from the coupon collector argument (cf., e.g., Theorem 1.23 in [3]) that, with probability at least $1 - n^{-1}$, there exists for each $i \in [n]$ at least one $t_i \leq 2n\log n$ such that $x^{(1,0)}$ and $x^{(1,t_i)}$ differ exactly in the i-th bit. Using this information, Lemma 10 yields that for each string $x^{(1,\ell)} \in \{x^{(1,t)} \mid t \in [0..2n\log n]\}$ with the largest fitness $f(x^{(1,\ell)}) = \max\{f(x^{(1,t)}) \mid t \in [0..2n\log n]\}$ it holds that $w_{\max} \in \mathcal{I}_{\max}(x^{(1,\ell)})$. Let us fix one such ℓ and set $x := x^{(1,\ell)}$.

Lemma 10 and Lemma 11 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 part of the first phase we aim at creating a string x' with $\mathcal{I}_{\max}(x') = \mathcal{I}$. We do that by querying $y =$ `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 $2n\log n$ iterations we have $\mathcal{I}_{\max}(x) = \mathcal{I}$, with probability at least $1 - n^{-1}$.

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

Learning instance $\mathcal{I}$. For the correctness of the sec-

ond phase observe that either we have $w_{\max} \geq \sum_{w \in \mathcal{I}} w/2$ in which case one of the strings $x^{(2,t)}$, $t \in [2n \log n]$ is optimal (i.e., $\{w_{\max}\} = \mathcal{I}_{\max}(x^{(2,t)})$ for some $t \in [2n \log n]$) with probability at least $1 - n^{-1}$. This is again the coupon collector argument. Note that we are done in this case. Otherwise we have that for any $t \leq 2n \log n$ 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 size $(f(x) - f(x^{(2,t)}))/2$. In this case we have, again by the coupon collector argument that $\mathcal{I}' := \{(f(x) - f(x^{(2,t)}))/2 \,|\, t \in [2n \log n]\}$ equals $\mathcal{I}$, with probability at least $1 - n^{-1}$.

Optimization phase. Knowing instance $\mathcal{I}$, we can now compute 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 9 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 $2n \log n$ 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 $n \log n$ iterations from below by $1 - (1 - 1/n)^{n \log n} \geq 1 - 1/n$. 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 8. We use the same notation as above.

Remark 9. *Let $\mathcal{I}$ be an instance of* Partition$_{\neq}$ *equipped with the ordering $\sigma_{\mathcal{I}}$ and fitness 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 follows from the second. But for a simple example consider the following situation. Let $\mathcal{I} := \{1, 2, 3, 4, 6\}$, $\sigma_{\mathcal{I}}$ the ordering of $\mathcal{I}$ and $x := (1, 0, 0, 0, 1)$, 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 fitness $f(y) = |6 - 10| = 4 = f(z)$. Hence, knowing x, knowing that $|x - z|_1 = 1$, and knowing the fitness values $f(x)$ and $f(z)$ does not suffice to compute the bit in which x and z differ.

For the second statement we distinguish whether $f(y) > f(x)$ or $f(x) > f(y)$.

Case 1, $f(y) > f(x)$**.** If $\mathcal{I}_{\max}(y) \cap \mathcal{I}_{\max}(x) \neq \{w_i\}$ then $w_i \notin \mathcal{I}_{\max}(x)$ for otherwise $f(y) = f(x) - 2w_i < f(x)$. Thus, $w_i \notin \mathcal{I}_{\max}(x)$ and $f(y) = f(x) + 2w_i$.

In case $\mathcal{I}_{\max}(y) \cap \mathcal{I}_{\max}(x) = \{w_i\}$ then $w_i \in \mathcal{I}_{\max}(x)$ and $f(y) = 2w_i - f(x)$. Furthermore, since $f(y) > f(x)$, this yields $w_i > f(x)$.

Case 2, $f(y) < f(x)$**.** In this case we must have $w_i \in \mathcal{I}_{\max}(x)$ for otherwise $f(y) = f(x) + 2w_i > f(x)$.

If $\mathcal{I}_{\max}(y) \subseteq \mathcal{I}_{\max}(x)$ then $f(y) = f(x) - 2w_i$ and if $\mathcal{I}_{\max}(y) \cap \mathcal{I}_{\max}(x) = \{w_i\}$ then $w_i > f(x)/2$ and $f(y) = 2w_i - f(x)$. Since $f(y) < f(x)$ we also have $w_i < f(x)$.

This enumerates all possible combinations and the claim follows. □

Lemma 10. *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 $f(x^{(\ell)}) = \max\{f(x^{(t)}) \,|\, t \in [0..n]\}$, 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 σ 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 ℓ imply $w_{\max} \notin \mathcal{I}_{\max}(x^{(0)})$ 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 11. *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 ℓ as in Lemma 10 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

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

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 ℓ. Thus, $f(x^{(\ell)}) \geq w_{\max}$. $\square$

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 4 solves PARTITION$_{\neq}$ and even PARTITION in a linear number of steps, using only unbiased variation operators of arity at most 3.

REMARK 12. *The 3-ary unbiased black-box complexity of* PARTITION *is at most linear in the size $|\mathcal{I}|$ of the input set $\mathcal{I}$.*

Since in the unrestricted model we can learn the weights by querying first the all-zeros bit string and then the n different unit vectors $(0, \dots, 0, 1, 0, \dots, 0)$, it is easy to verify the following.

THEOREM 13. *The unrestricted black-box complexity of* PARTITION *is at most linear in the size $|\mathcal{I}|$ of the input set $\mathcal{I}$.*

5. CONCLUSIONS

We have shown that already the unary unbiased black-box model contains algorithms which solve $\mathcal{NP}$-hard problems in a polynomial number of queries. Furthermore, we could also prove that the unary unbiased black-box complexity of the JUMP$_k$ function is of order $O(n \log n)$, a bound which is not achieved by standard search heuristics.

These results indicate that the unbiased black-box model, while clearly closer to the truth than the unrestricted one, still does not give a full picture on how difficult a problem is to be solved 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 and the third author [7], 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 value of the fitness. We do not know yet what is the black-box complexity of, e.g., PARTITION in this new model.

Acknowledgment

We would like to thank the anonymous reviewers for numerous comments that significantly improved the presentation of this work.

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

Carola Winzen is a recipient of the Google Europe Fellowship in Randomized Algorithms, and this research is supported in part by this Google Fellowship.

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

6. REFERENCES

[1] D. Aldous. Minimization algorithms and random walk on the d-cube. *Annals of Probability*, 11:403–413, 1983.

[2] G. Anil and R. P. Wiegand. Black-box search by elimination of fitness functions. In *Proc. of Foundations of Genetic Algorithms (FOGA'09)*, pages 67–78. ACM, 2009.

[3] A. Auger and B. Doerr. *Theory of Randomized Search Heuristics*. World Scientific, 2011.

[4] E. G. Coffman and W. Whitt. Recent asymptotic results in the probabilistic analysis of schedule makespans. In P. Chrétienne, E. G. Coffman, J. K. Lenstra, and Z. Liu, editors, *Scheduling Theory and Its Applications*, pages 15–31. Wiley, 1995.

[5] B. Doerr, D. Johannsen, T. Kötzing, P. K. Lehre, M. Wagner, and C. Winzen. Faster black-box algorithms through higher arity operators. In *Proc. of Foundations of Genetic Algorithms (FOGA'11)*, pages 163–172. ACM, 2011.

[6] B. Doerr, T. Kötzing, J. Lengler, and C. Winzen. Black-Box Complexities of Combinatorial Problems. In *Proc. of Genetic and Evolutionary Computation Conference (GECCO'11)*. ACM, 2011.

[7] B. Doerr and C. Winzen. Towards a Complexity Theory of Randomized Search Heuristics: Ranking-Based Black-Box Complexity. *ArXiv e-prints 1102.1140*, 2011. To appear in Proc. of Computer Science Symposium in Russia (CSR'11), Springer.

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

[9] S. Droste, T. Jansen, and I. Wegener. Upper and lower bounds for randomized search heuristics in black-box optimization. *Theoretical Computer Science*, 39:525–544, 2006.

[10] J. B. G. Frenk and A. H. G. R. Kan. The rate of convergence to optimality of the LPT rule. *Discrete Applied Mathematics*, 14:187–197, 1986.

[11] M. R. Garey and D. S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., 1990.

[12] D. Hausmann and B. Korte. Lower bounds on the worst-case complexity of some oracle algorithms. *Discrete Mathematics*, 24:261–276, 1978.

[13] J. Hromkovič. *Algorithmics for hard problems: introduction to combinatorial optimization, randomization, approximation, and heuristics*. Springer-Verlag New York, Inc., 2nd edition, 2003.

[14] R. M. Karp. Reducibility Among Combinatorial Problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

[15] P. K. Lehre and C. Witt. Black-box search by unbiased variation. In *Proc. of Genetic and Evolutionary Computation Conference (GECCO'10)*, pages 1441–1448. ACM, 2010.

[16] D. C. Llewellyn, C. Tovey, and M. Trick. Local optimization on graphs. *Discrete Applied Mathematics*, 23:157–178, 1989.

[17] C. Witt. Worst-case and average-case approximations by simple randomized search heuristics. In *Proc. of Symposion on Theoretical Aspects of Computer Science (STACS'05)*, pages 44–56. Springer, 2005.