Black-Box Complexity of the Binary Value Function

Nina Bulanova
ITMO University
Saint Petersburg, Russia
ninasbulanova@gmail.com

Maxim Buzdalov
ITMO University
Saint Petersburg, Russia
mbuzdalov@gmail.com

ABSTRACT

The binary value function, or BinVal, has appeared in several studies in theory of evolutionary computation as one of the extreme examples of linear pseudo-Boolean functions. Its unbiased black-box complexity was previously shown to be at most $\lceil \log_2 n \rceil + 2$, where n is the problem size.

We augment it with an upper bound of $\log_2 n + 2.42141558 - o(1)$, which is more precise for roughly a half of values of n. We also present a lower bound of $\log_2 n + 1.1186406 - o(1)$ and provide an algorithm to compute the exact black-box complexity of BinVal for a given n.

CCS CONCEPTS

• **Theory of computation → Theory of randomized search heuristics**.

KEYWORDS

Unbiased black-box complexity, linear functions, BinVal.

ACM Reference Format:
Nina Bulanova and Maxim Buzdalov. 2019. Black-Box Complexity of the Binary Value Function. In *Genetic and Evolutionary Computation Conference Companion (GECCO '19 Companion), July 13–17, 2019, Prague, Czech Republic.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3319619.3322070

1 INTRODUCTION AND DEFINITIONS

In the current state of theory of randomized search heuristics there are two major building blocks that augment each other: runtime analysis and black-box complexity theory. The gaps between their results are an important source of difficult questions and new inspiring results.

The black-box complexity is defined for a class of optimization problems (or functions) F and a class of algorithms $\mathcal{X}$. We denote by $E_{\mathcal{A}}(f)$ the expected running time of an algorithm $\mathcal{A}$ on a function f, that is, the expected number of queries to that function until its optimum is queried for the first time. The *black-box complexity* of a class of functions F for a class of algorithms $\mathcal{X}$ is [3]:

$$BBC_{\mathcal{X}}(F) = \inf_{\mathcal{A} \in \mathcal{X}} \sup_{f \in F} E_{\mathcal{A}}(f).$$

In the *unrestricted* black-box complexity, as defined in [4], the set $\mathcal{X}$ includes all possible black-box algorithms. To get more realistic complexities, the unbiased black-box complexity was introduced in [5] for pseudo-Boolean functions by restricting the possible operations to those typically performed by evolutionary algorithms, and [6] extends unbiasedness to arbitrary functions and shows that, with a proper definition, the unbiased black-box complexity coincides with the unrestricted one. The works [1, 3] study a further restriction, called the k-ary unbiased black-box complexity, which allows using only k-ary unbiased variation operators.

A *k-ary variation operator* [3] $\mathcal{R}$ produces a search point y from the k search points $x_1, \ldots, x_k$ with probability $P_{\mathcal{R}}(y \mid x_1, \ldots, x_k)$. The operator $\mathcal{R}$ is *unbiased* [5] if the following holds for all search points $x_1, \ldots, x_k, y, z$ and all permutations π of size n:

$$P_{\mathcal{R}}(y \mid x_1, \ldots, x_k) = P_{\mathcal{R}}(y \oplus z \mid x_1 \oplus z, \ldots, x_k \oplus z),$$
$$P_{\mathcal{R}}(y \mid x_1, \ldots, x_k) = P_{\mathcal{R}}(\pi(y) \mid \pi(x_1), \ldots, \pi(x_k)),$$

where $a \oplus b$ is the bitwise exclusive-or operation and $\pi(a)$ is an application of permutation π to bit string a.

One result from [1] is that the binary unbiased black-box complexity is $O(n)$ for OneMax, supported by an algorithm with the expected runtime of $2n$. Our conjecture is that the binary unbiased black-box complexity is linear not only for OneMax, but for any linear function, as the mentioned algorithm works on linear functions without any changes. The class of linear functions includes the BinVal function, a linear function with weights equal to powers of two, which is defined on bit strings of length n as follows [2]:

$$\text{BinVal}_{z,\pi}(x) = \sum_{i=1}^{n} 2^{i-1} \cdot [z_i = x_{\pi(i)}],$$

where $z \in \{0; 1\}^n$ is the (unknown) optimum, and π is a hidden permutation of size n that defines which weights are given to which indices. This function has a single global maximum at $x = z$ with the corresponding function value of $2^n - 1$. In [2], its black-box complexity was proven to be at most $\lceil \log_2 n \rceil + 2$.

We feel that BinVal is a promising tool for analyzing k-ary unbiased black-box complexities. For this reason, in this paper we prove quite sharp bounds on its unbiased black-box complexity. Our upper bound of $\log_2 n + 2.42141558 - o(1)$ complements the existing bound and is more precise for roughly a half of possible n. We also prove a lower bound of $\log_2 n + 1.1186406 - o(1)$ for the first time. Finally, we present the algorithm that evaluates the exact complexity value for any given n.

This research was supported by the Russian Scientific Foundation, agreement No. 17-71-20178. This paper omits all proofs for space reasons; the larger version of this paper with the missing proofs and more detailed explanations is available at arXiv[1].

[1]https://arxiv.org/abs/1904.04867

2 BLACK-BOX COMPLEXITY OF BINVAL

We start with sketching an algorithm that works in $O(\log n)$. From the result $\text{BinVal}(x_0)$ of the initial query x_0, which is essentially a random bit string, we can find the set of weights, at which the bits are guessed correctly, but not yet their positions. By issuing the next query x_1 where a random half of the bits is flipped and analyzing $\text{BinVal}(x_1)$, we get which weights correspond to bits which coincide in x_0 and x_1, and which correspond to differing bits, but nothing more. These halves, or subproblems, are two functions which are identical to BinVal of a smaller size (except that the weights are not contiguous powers of two, which changes nothing).

We can optimize the subproblems in parallel by combining the queries coming from the subproblems into a single query to the original problem, and splitting an answer into the answers to the queries of the subproblems. On a next step, each of these two problems is again subdivided into halves, and this process continues until the subproblem sizes approach one. Some subproblems will be solved earlier by occasionally making all bits equal one or zero. As we know the number of bits guessed right, we may derive an optimal decision on how to split the problem into two subproblems.

We formalize these ideas using the following set of statements.

Definition 2.1. $E(n, d)$, where $0 \le d \le n$, is the expected time to optimize a uniformly sampled from problem of size n from the BinVal class, using an optimal algorithm, where only the first query has been made and its Hamming distance to the optimum is d.

It is clear that $E(n, 0) = 0$, since the first query has already queried the optimum, and $E(n, n) = 1$, as the optimum is the complete inverse of the first query, so for $n = 1$ all possible values are already known. The following lemma helps to derive all other values.

Lemma 2.2. *The following holds for $n > 1$ and $0 < d < n$:*

$$E(n,d) = 1 + \min_{0<s<n} E(n,d,s), \text{ where}$$

$$E(n,d,s) = \sum_{t=\max(0,s+d-n)}^{\min(s,d)} \max(E(s,s-t), E(n-s,d-t)) \cdot \frac{\binom{s}{t}\binom{n-s}{d-t}}{\binom{n}{d}}.$$

With this lemma we can constrain $E(n, d)$ quite tight.

Lemma 2.3. *For all $n > 0$ and all $0 < d < n$ it holds that:*

$$E(n,d) \le \log_2 n + 1 + \sum_{z=0}^{\lceil \log_2 n \rceil - 1} \log_2 \left(1 + \frac{1}{2^z}\right).$$

The sum above is upper-bounded by a constant:

$$\sum_{z=0}^{\infty} \log_2 \left(1 + \frac{1}{2^z}\right) = 2.2535240379347\ldots \le 2.26,$$

so it follows from Lemma 2.3 that $E(n, d) \le \log_2 n + 3.26$. We refine the additive constant by evaluating all $E(n, d)$ by definition for all $n \le 2^k$, computing the maximum difference $D_{\max} = E(n, d) - \log_2 n$, and adding the following to $D_{\max}$:

$$\sum_{z=k}^{\infty} \log_2 \left(1 + \frac{1}{2^z}\right).$$

For $k = 10$ we found that $D_{\max} < 1.4194631$, and the analytical remainder converges to a number slightly smaller than 0.00195248, which together proves that $E(n, d) \le \log_2 n + 1.42141558$.

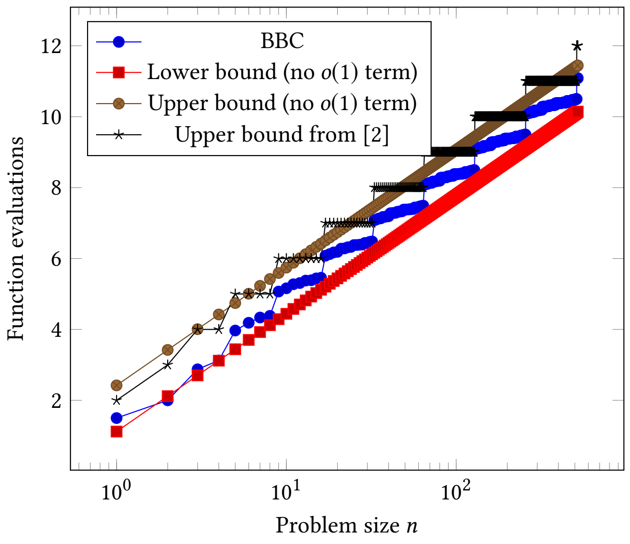

Figure 1: Plots of the exact black-box complexity of BinVal and of its upper and lower bounds

We also prove the lower bounds.

Lemma 2.4. *For all $n > 0$ and $0 < d < n$ it holds that:*

$$E(n,d) \ge \log_2 n + 0.1186406.$$

In practice, due to various pessimizations in our proofs, the lower bound is a little bit better: $E(n, d) \ge \log_2 + \frac{1}{6}$ for all n and $0 < d < n$. The main result of our paper follows from Lemmas 2.3 and 2.4.

Theorem 2.5. *The black-box complexity of BinVal is:*

$$\text{at most } \log_2 n + 2.42141558 - \Theta(\log_2 n/2^n),$$
$$\text{at least } \log_2 n + 1.1186406 - \Theta(\log_2 n/2^n).$$

3 CONCLUSION

We proved quite sharp bounds on the black-box complexity of the binary value function, which is $\log_2 n + \Theta(1)$, where the constants that define $\Theta(1)$ were also found and their difference is less than 1.4. The upper bound complements the existing upper bound from [2], as it is more precise for roughly a half of problem sizes (see Fig. 1 for visual comparison), and the lower bound was proven for the first time. We hope that this result will be useful to close the existing gaps in k-ary unbiased black-box complexities.

REFERENCES

[1] Benjamin Doerr, Daniel Johannsen, Timo Kötzing, Per Kristian Lehre, Markus Wagner, and Carola Winzen. 2011. Faster black-box algorithms through higher arity operators. In *Proceedings of Foundations of Genetic Algorithms*. 163–172.
[2] Benjamin Doerr and Carola Winzen. 2014. Ranking-Based Black-Box Complexity. *Algorithmica* 68, 3 (2014), 571–609.
[3] Benjamin Doerr and Carola Winzen. 2014. Reducing the arity in unbiased black-box complexity. *Theoretical Computer Science* 545 (2014), 108–121.
[4] Stefan Droste, Thomas Jansen, and Ingo Wegener. 2006. Upper and Lower Bounds for Randomized Search Heuristics in Black-Box Optimization. *Theory of Computing Systems* 39, 4 (2006), 525–544.
[5] Per Kristian Lehre and Carsten Witt. 2012. Black-box Search by Unbiased Variation. *Algorithmica* 64 (2012), 623–642.
[6] Jonathan Rowe and Michael Vose. 2011. Unbiased Black Box Search Algorithms. In *Proceedings of Genetic and Evolutionary Computation Conference*. ACM, 2035–2042.