An Algorithm for Computing Lower Bounds for Unrestricted Black-Box Complexities

Maxim Buzdalov
ITMO University
49 Kronverkskiy ave.
Saint-Petersburg, Russia
mbuzdalov@gmail.com

ABSTRACT

Finding and proving lower bounds on black-box complexities is one of the hardest problems in theory of randomized search heuristics. Until recently, there were no general ways of doing this, except for information theoretic arguments similar to the one of Droste, Jansen and Wegener.

In a recent paper by Buzdalov, Kever and Doerr, a theorem is proven which may yield tighter bounds on unrestricted black-box complexity using certain problem-specific information. To use this theorem, one should split the search process into a finite number of states, describe transitions between states, and for each state specify (and prove) the maximum number of different *answers* to any query.

We augment these state constraints by one more kind of constraints on states, namely, the maximum number of different currently possible *optima*. An algorithm is presented for computing the lower bounds based on these constraints. We also empirically show improved lower bounds on black-box complexity of OneMax and Mastermind.

Keywords

Black-box complexity, lower bounds, OneMax, Mastermind.

1. INTRODUCTION AND MAIN IDEAS

Black-box complexity of an optimization problem is the smallest expected number of queries a black-box search algorithm needs to solve the problem, that is, to query one of its optima. Various kinds of black-box complexities can be studied: the *unrestricted* black-box complexity [4] allows any kind of black-box search algorithms to be used, while, for example, the *unbiased* black-box complexity [6] restricts the set of allowed algorithms to the ones using only unbiased variation operators. Comparing black-box complexities of certain problem classes with the running times[1] of existing randomized search heuristics helps to understand how good they are and sometimes to construct better algorithms [3].

In this paper, we restrict ourselves to lower bounds for unrestricted black-box complexities. Until recently, there was only one general purpose method for constructing such lower bounds based on information theoretic arguments [4, Theorem 2]. This method uses the Yao's minimax principle [7] to reduce the lower bound analysis of randomized algorithms to the lower bound analysis of deterministic algorithms. For the latter, the theorem suggests to prove, based on the problem's properties, the upper limit on the number of possible answers to each query by some number $k \geq 2$. This limits the breadth of the decision tree of any deterministic algorithm solving this problem. If every element of the search space S is an optimum of exactly one problem instance, this theorem proves a lower bound of $\lceil \log_k |S| \rceil - 1$.

In 2015, an extension of this theorem was presented and proven [1, Theorem 7]. The idea of this extension is based on distinguishing several types of decision tree nodes of algorithms solving the considered problem and proving, for each type i separately, the maximum number A_{ij} of different answers, after which the next decision tree node has the type j. This theorem is able to prove stronger lower bounds [1].

In the current research, we augment the description of the problem, which consists of describing transitions between decision tree node types as in [1], by proving the maximum possible number of different optima B_i in a subtree of a decision tree node of a type i for every possible i, which should improve the lower bounds even further. Although tracking the maximum possible number of optima is known since [5], combining it with typed decision tree nodes is new.

We are not yet ready to present a theorem similar to Theorem 7 from [1] which would allow to prove lower bounds based on the knowledge, for all problem sizes, of the matrix $A = A_{ij}$ and the vector $B = B_i$. However, to test if this concept is useful to prove stronger bounds, we implemented an algorithm which evaluates the lower bound given the matrix A and the vector B. Apart from this algorithm (which can be used by researchers to check if a certain way of building A and B helps proving stronger bounds), we show some promising empirical results for the well-known OneMax problem, as well as for the Mastermind problem [2].

2. ALGORITHM OUTLINE

We briefly describe the proposed algorithm for finding the lower bound on the problem's unrestricted black-box complexity for a certain problem size, given the node type interaction matrix A and the vector B of maximum numbers

[1] A *running time* of a black-box algorithm is the number of queries it did until an optimum is hit for the first time.

GECCO '16 July 20-24, 2016, Denver, CO, USA

ACM ISBN 978-1-4503-4323-7/16/07.

DOI: http://dx.doi.org/10.1145/2908961.2908986

n	Simple	Simple cfg		Complex cfg		Simple
	lower	Value	Time	Value	Time	upper
1000	100.999	112.927	0.000	121.174	2.031	200.687
2000	183.000	201.982	0.000	215.713	23.93	364.771
3000	260.000	285.833	0.002	303.867	84.01	519.447
4000	335.000	365.994	0.001	388.440	291.8	668.573
5000	407.000	443.999	0.002	470.649	822.1	813.821

Table 1: Results for OneMax

n	Simple	Simple cfg	
	lower	Value	Time
100000	6021.00	6407.92	0.632
200000	11358.0	12042.8	2.233
300000	16489.0	17449.0	5.474
400000	21495.0	22716.0	8.860
500000	26411.0	27885.0	13.84

Table 2: Results for OneMax using the simple configuration and large n

n	Simple cfg		Complex cfg		Ratio
	Value	Time	Value	Time	$r - 8/7$
100	100.990	0.003	121.456	0.114	+0.0717
200	200.995	0.001	238.452	0.512	+0.0494
300	300.997	0.001	354.271	2.165	+0.0380
400	400.997	0.002	469.219	5.518	+0.0302
500	500.998	0.001	583.490	12.67	+0.0241
600	600.998	0.002	697.511	26.67	+0.0197
700	700.999	0.003	811.272	46.08	+0.0161
800	800.999	0.004	924.819	79.58	+0.0132
900	900.999	0.004	1038.09	118.9	+0.0106
1000	1001.00	0.005	1151.11	183.9	+0.0083
1100	1101.00	0.006	1264.04	283.4	+0.0063
1200	1201.00	0.007	1376.72	428.2	+0.0044
1300	1301.00	0.009	1489.27	619.6	+0.0027
1400	1401.00	0.012	1601.89	890.9	+0.0014
1500	1501.00	0.012	1714.17	1233	−0.0001

Table 3: Results for Mastermind

of different optima in a node subtree. The algorithm implementation is available at GitHub[2].

The brief idea of the algorithm is to lazily build an infinite tree defined by the matrix A and fill it with the problem instances, paying attention to the constraints of B. A naive implementation would take roughly $O(|S| \cdot D)$ time, where S is the search space and D is the average depth of a problem instance (that is, the answer). While D is often polynomial in the problem size, $|S|$ is typically exponential. We reduce the running time using the following ideas.

First, multiple arcs from a node to nodes of the same type can be processed at once, as the corresponding subtrees and the size constraints are identical. Second, different nodes of the same type will, in fact, accept different number of problem instances, depending on the nodes above. However, they behave the same until the constraint is hit for this node or for any of the nodes above. The algorithm tracks how much problem instances a subtree of an "ideal" node of type t contains at level d for all possible t and d. For all "real" nodes, these values will be the same for all $d < d'$, where d' depends on the actual node. Ideal and real nodes are interdependent (e.g. the children of an ideal node are real nodes), but when depths are taken into account, no circular dependencies arise, so lazy evaluation resolves all the issues.

The running time of the algorithm as $O(|B|^2 D(\log S)^2)$, where D is the answer. Although it is still linear in D, it is no more polynomial in the search space size S.

3. ONEMAX AND MASTERMIND

For both problems OneMax and Mastermind we developed two ways, which we call *configurations*, of generating the matrix A and the vector B from the problem size N.

The *simple configurations* for both problems have only two types, type 1 for the root node and type 2 for all other nodes. Type 2 nodes have B_2 equal to the maximum possible number of different optima after receiving an answer to any query: $\binom{n}{\lfloor n/2 \rfloor}$ for OneMax, $n \cdot (n-1)^{n-1}$ for Mastermind.

The *complex configurations* for both problems have $\Theta(n)$ types which, roughly speaking, differentiate between receiving small, medium and large answers. For OneMax, both small and large answers limit the number of optima severely, while for Mastermind only larger answers do.

For OneMax, the results are shown in Table 1 along with the running times of the algorithm. Small running times for the simple configuration motivated us to run it for much larger problem sizes. These results are shown in Table 2. The new lower bounds are noticeably larger than the known ones even for the simple configuration, and are even larger for the complex configuration. The values of the simple configuration fit, up to an additive constant not exceeding 2, to:

$$\frac{n}{\log_2 n} + \frac{n}{(\log_2 n)^2} + \frac{n}{(\log_2 n)^3} + \ldots = \frac{n}{\log_2 n} \cdot \frac{1}{1 - (\log_2 n)^{-1}}.$$

For Mastermind, the results are shown in Table 3. Here, the complex configuration seems to improve the lower bound by a constant factor close to 8/7.

[2]https://github.com/mbuzdalov/papers/tree/master/2016-gecco-bbcomp-algo

4. REFERENCES

[1] M. Buzdalov, M. Kever, and B. Doerr. Upper and lower bounds on unrestricted black-box complexity of $\textsc{Jump}_{n,\ell}$. In *Evolutionary Computation in Combinatorial Optimization*, number 9026 in Lecture Notes in Computer Science, pages 209–221. 2015.

[2] V. Chvátal. Mastermind. *Combinatorica*, 3(3):325–329, 1983.

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

[4] S. Droste, T. Jansen, and I. Wegener. Upper and lower bounds for randomized search heuristics in black-box optimization. *Theory of Computing Systems*, 39(4):525–544, 2006.

[5] D. E. Knuth. The computer as Master Mind. *Journal of Recreational Mathematics*, 9:1–6, 1976.

[6] P. K. Lehre and C. Witt. Black-box search by unbiased variation. In *Proceedings of Genetic and Evolutionary Computation Conference*, pages 1441–1448. ACM, 2010.

[7] A. C.-C. Yao. Probabilistic computations: Toward a unified measure of complexity. In *18th Annual Symposium on Foundations of Computer Science*, pages 222–227, 1977.