Upper and Lower Bounds on Unrestricted Black-Box Complexity of $\textsc{Jump}_{n,\ell}$

Maxim Buzdalov[1], Mikhail Kever[1], and Benjamin Doerr[2]

[1] ITMO University, 49 Kronverkskiy av.,
Saint-Petersburg, Russia, 197101
mbuzdalov@gmail.com, mikhail.kever@gmail.com
[2] LIX, École polytechnique, 91128 Palaiseau Cedex
doerr@lix.polytechnique.fr

Abstract. We analyse the unrestricted black-box complexity of $\textsc{Jump}_{n,\ell}$ functions. For upper bounds, we present three algorithms for small, medium and extreme values of ℓ. We present a matrix lower bound theorem which is capable of giving better lower bounds than a general information theory approach if one is able to assign different types to queries and define relationships between them. Using this theorem, we prove lower bounds for Jump separately for odd and even values of n. For several cases, notably for extreme Jump, the first terms of lower and upper bounds coincide.

1 Introduction

To understand how evolutionary algorithms (and other black-box optimizers as well) behave when optimizing certain functions, it is possible to construct upper bounds (by constructing and studying various algorithms), as well as lower bounds (by studying how fast an algorithm can be in principle), which complement each other. Comparing these bounds allows to evaluate how good today's heuristics are and sometimes to construct better algorithms by learning from black-box [1].

Black-box complexity studies how many function evaluations are needed in expectation by an optimal black-box algorithm until it queries an optimum for the first time. As randomized search heuristics are black-box optimizers, black-box complexity of a problem gives a lower bound on the number of fitness evaluations of any search heuristic to solve this problem.

In this paper we consider optimization of functions mapping bit strings of fixed length to integers — the *pseudo-Boolean* functions. A famous class of such functions is OneMax — having a certain hidden bit string z of length n, for a bit string x of length n the function $\textsc{OneMax}_{n,z}(x)$ returns the number of bits coinciding both in x and z. Jump, another popular class of functions, takes another parameter ℓ and zeroes out the values of OneMax for every string except z that are at the distance of at most ℓ from both z and its inverse.

A more formal definition of JUMP is as follows:

$$\textsc{Jump}_{n,\ell,z} = \begin{cases} n & \text{if } \textsc{OneMax}_{n,z} = n \\ \textsc{OneMax}_{n,z} & \text{if } \ell < \textsc{OneMax}_{n,z} < n - \ell \\ 0 & \text{otherwise.} \end{cases}$$

Most of times, when z does not matter, we write just $\textsc{OneMax}_n$ and $\textsc{Jump}_{n,\ell}$. The special case of $\ell = \left\lfloor \frac{n}{2} \right\rfloor - 1$, which is the maximum possible ℓ that doesn't zero out the middle fitness values, is called *extreme* JUMP.

In this paper, we consider *unrestricted* black-box complexity (which was introduced in Droste et al [4]) of the $\textsc{Jump}_{n,\ell}$ problem. Another kind of black-box complexity, the *unbiased* black-box complexity, was considered for JUMP in Doerr et al [2].

The rest of the paper is structured as follows. Section 2 is dedicated to the upper bounds on JUMP which are proven by giving the corresponding algorithms and discussing their complexity. In Section 3, the *matrix lower bound theorem*, which is somewhat similar to Theorem 2 from [4] but is able to produce better lower bounds, is described and proven. Section 4 describes lower bounds on JUMP which are constructed from the matrix theorem. Section 5 concludes.

2 Upper Bounds for $\textsc{Jump}_{n,\ell}$

Here, the upper bounds for JUMP are considered. In Section 2.1 several useful helper theorems are referenced or proven. Section 2.2 is dedicated to smaller ℓ, Section 2.3 is for larger ℓ, and Section 2.4 considers the case of extreme JUMP.

2.1 Helper Theorems

Theorem 1. *For sufficiently large n, for $t \geq \left(1 + \frac{4 \log_2 \log_2 n}{\log_2 n}\right) \frac{2n}{\log_2 n}$ and for an even $d \in [2; n]$ it holds that $\binom{n}{d} \left(\binom{d}{d/2} 2^{-d} \right)^t \leq 2^{-3t/4}$.*

Proof. This is proven in Doerr et al [3] as Statement 8. □

Theorem 2. *For sufficiently large n, for $\ell < n/2 - \sqrt{n} \log_2 n$ and for $x \in \{0,1\}^n$ taken uniformly at random the probability for $\textsc{Jump}_{n,\ell}(x)$ to be zero is at most $2e^{-2(\log_2 n)^2}$.*

Proof. The value of $\textsc{OneMax}(x)$ for a random x has a binomial distribution with parameters n and $p = 1/2$. From Hoeffding's inequality [5], for $k \leq np$, the distribution function for binomial distribution $F_{n,p}(k)$ is bound from above by $e^{-2\frac{(np-k)^2}{n}}$. As a consequence, the probability for $\textsc{Jump}_{n,\ell}(x)$ to be zero is at most $2F_{n,1/2}(l) \leq 2e^{-2\frac{(n/2-l)^2}{n}} \leq 2e^{-2\frac{\left(\sqrt{n}\log_2 n\right)^2}{n}} = 2e^{-2(\log_2 n)^2}$. □

Theorem 3. *Assume that n is sufficiently large and $\ell < n/2 - \sqrt{n}\log_2 n$. Let $z \in \{0,1\}^n$, and X be a set of $t \geq \left(1 + \frac{4\log_2\log_2 n}{\log_2 n}\right)\frac{2n}{\log_2 n}$ elements from $\{0,1\}^n$ chosen randomly using uniform distribution and mutually independently. The probability that there exists an $y \in \{0,1\}^n$ such that $y \neq z$ and $\textsc{Jump}_{n,\ell,z}(x) = \textsc{Jump}_{n,\ell,y}(x)$ for all $x \in X$, is at most $2^{-t/4}$.*

Proof. Let's define A_d as a set of points which differ from z in exactly d positions where $0 \leq d \leq n$.

We say that a point $y \in \{0,1\}^n$ *agrees* with $x \in X$ if $\textsc{Jump}_{n,\ell,z}(x) = \textsc{Jump}_{n,\ell,y}(x)$. This means that either $\textsc{Jump}_{n,\ell,z}(x) = 0$ or $\textsc{OneMax}_{n,y}(x) = \textsc{OneMax}_{n,z}(x)$. The probability of the former does not exceed $2e^{-2(\log_2 n)^2}$ by Theorem 2. The latter holds iff x and y (as well as x and z) differ in exactly half of the d bits in which y and z differ. To sum up, if $y \in A_d$, the probability for y to agree with a random x is at most $2e^{-2(\log_2 n)^2}$ for an even d and at most $2e^{-2(\log_2 n)^2} + \binom{d}{d/2}2^{-d}$ for an odd d. As for large enough n it holds that $2e^{-2(\log_2 n)^2} \leq \left(2^{1/4} - 1\right)\binom{d}{d/2}2^{-d}$, the latter is at most $2^{1/4}\binom{d}{d/2}2^{-d}$.

Let p be the probability that there exists an $y \in \{0,1\}^n \setminus \{z\}$ such that y agrees with all $x \in X$. Then the following holds:

$$
\begin{aligned}
p &= Pr\left(\bigcup_{y\in\{0,1\}^n\setminus\{z\}} \bigcap_{x\in X} y \text{ agrees with } x\right) \\
&\leq \sum_{y\in\{0,1\}^n\setminus\{z\}} Pr\left(\bigcap_{x\in X} y \text{ agrees with } x\right) \\
&= \sum_{d=1}^{n}\sum_{y\in A_d}\prod_{x\in X} Pr(y \text{ agrees with } x) \\
&\leq \sum_{d \text{ even}}\binom{n}{d}\left(2^{1/4}\binom{d}{d/2}2^{-d}\right)^t + \sum_{d \text{ odd}}\binom{n}{d}\left(2e^{-2(\log_2 n)^2}\right)^t \\
&= \sum_{d \text{ even}}\binom{n}{d}\left(2^{1/4}\binom{d}{d/2}2^{-d}\right)^t + 2^{n-1}\left(2e^{-2(\log_2 n)^2}\right).
\end{aligned}
$$

After applying Theorem 1, we get that:

$$p \leq \frac{n+1}{2}2^{t/4}2^{-3t/4} + 2^{n-1+t}e^{-2t(\log_2 n)^2},$$

which is less than $2^{-t/4}$ for sufficiently large n. □

2.2 Upper Bound for Smaller ℓ

Theorem 4. *If $\ell < n/2 - \sqrt{n}\log_2 n$, the unrestricted black-box complexity of $\textsc{Jump}_{n,\ell}$ is at most $(1 + o(1))\frac{2n}{\log_2 n}$, where $o(1)$ is measured relative to n.*

Proof. We use the same algorithm which is used in [3] for proving the lower bound for OneMax. We select randomly and independently t queries such that $t \geq \left(1 + \frac{4 \log_2 \log_2 n}{\log_2 n}\right) \frac{2n}{\log_2 n}$ and check if there exists a single optimum z which agrees with all these queries (a query q with an answer a agrees with an optimum z if $\textsc{Jump}_{n,\ell,z}(q) = a$). The complexity of one iteration equals to t and the probability of not finding an optimum is at most $2^{-t/4}$ by Theorem 3. Thus the complexity of the algorithm is at most $\frac{t}{1-2^{-t/4}} = (1 + o(1)) \frac{2n}{\log_2 n}$. □

2.3 Upper Bound for Larger ℓ

For bigger l, the $\textsc{Jump}_{n,\ell}$ problem can be solved by reduction to smaller Jump problems for which the algorithm for the previous section suffices.

Theorem 5. *For $\frac{n}{2} - \sqrt{n} \log_2 n \leq \ell < \left\lfloor \frac{n}{2} \right\rfloor - 1$ the unrestricted black-box complexity of $\textsc{Jump}_{n,\ell}$ is at most $(1 + o(1)) \frac{n}{\log_2(n-2\ell)}$ where $o(1)$ is measured when $(n - 2\ell) \to \infty$.*

Proof. Assume that $k = \left\lfloor \frac{n}{2} \right\rfloor - \ell - 1 \neq 0$. We reduce our problem to $\textsc{Jump}_{s,\frac{s}{2}-k-1}$ where $\sqrt{s} \log_2 s < k$. The algorithm is outlined at Fig. 1

First the algorithm finds a maximum even s such that $\sqrt{s} \log_2 s < k$, which would allow applying Theorem 4 for solving $\textsc{Jump}_{s,\frac{s}{2}-k-1}$. After that, the algorithm finds a string $x \in \{0,1\}^n$ with exactly $\left\lfloor \frac{n}{2} \right\rfloor$ correct bits using random queries. The probability that $\textsc{Jump}_{n,\ell}$ is equal to $\left\lfloor \frac{n}{2} \right\rfloor$ for a random query is $2^{-n}\binom{n}{\lfloor n/2 \rfloor}$ which is $\Theta\left(\frac{1}{\sqrt{n}}\right)$ by Stirling's formula. This means that the string x can be found in $\Theta(\sqrt{n})$ queries.

After finding the x, the algorithm splits all bit indices into sets of size s (except for probably one) in such a way that in each set exactly half of bits coincide with those in the answer. This is done in lines 8–15 at Fig. 1, where b_i is the i-th such set. B, the set of yet undistributed bits, always contains indices of which exactly $|B|/2$ indices correspond to correctly guessed bits.

To do that, the algorithm generates random subsets of size s and checks each of them if it contains exactly $\frac{s}{2}$ correct bits, which is done by flipping the bits from the chosen subset and checking whether the fitness function returns $\left\lfloor \frac{n}{2} \right\rfloor$. If $|B| = m$, the probability of correct selection is:

$$p = \binom{\lfloor m/2 \rfloor}{s/2} \binom{\lceil m/2 \rceil}{s/2} \binom{m}{s}^{-1} = \frac{\lfloor m/2 \rfloor! \lceil m/2 \rceil! s! (m-s)!}{\lfloor (m-s)/2 \rfloor! \lceil (m-s)/2 \rceil! (s/2)!}$$

$$= \binom{s}{s/2} \binom{m-s}{\lfloor (m-s)/2 \rfloor} \binom{m}{\lfloor m/2 \rfloor}^{-1} = \Theta\left(\frac{\frac{2^s}{\sqrt{s}} \frac{2^{m-s}}{\sqrt{m-s}}}{\frac{2^m}{\sqrt{m}}} \right)$$

$$= \Theta\left(\frac{\sqrt{m}}{\sqrt{s}\sqrt{m-s}} \right) = \Omega\left(\frac{1}{\sqrt{s}} \right).$$

This gives an $O(\sqrt{s})$ bound for one subset selection and an $O(n/\sqrt{s})$ bound on entire process of finding subsets.

```
1: function LargeJump(n, ℓ, f ∈ Jump_{n,ℓ})
2:     k ← ⌊n/2⌋ − ℓ − 1
3:     s ← max{w | √w log_2 w < k; w is even}
4:     τ ← ⌈n/s⌉
5:     repeat
6:         x ← Uniform()
7:     until f(x) = ⌊n/2⌋
8:     B ← [1; n]
9:     for i ∈ [1; τ) do
10:        repeat
11:            b_i ← ChooseSubsetRandomlyUniformly(B, s)
12:        until f(FlipBits(x, b_i)) = ⌊n/2⌋
13:        B ← B \ b_i
14:    end for
15:    b_τ ← B
16:    ω_0 ← x
17:    for i ∈ [1; τ] do
18:        α_i ← SmallJumpProjection(b_i, x)
19:        ω_i ← SetAtPositions(b_i, ω_{i−1}, α_i)
20:    end for
21:    return ω_τ
22: end function
```

Fig. 1. Algorithm for $\textsc{Jump}_{n,\ell}$ with $\frac{n}{2} - \sqrt{n}\log_2 n \le \ell < \left\lfloor \frac{n}{2} \right\rfloor - 1$

Next, the algorithm optimizes separately bits from each of the subsets b_i using the algorithm for small Jump from Theorem 4 (lines 17–20 at Fig. 1). If every query for a subproblem on bits from b_i is forwarded to the main function f with all bits not from b_i taken from x, the resulting subproblem becomes exactly a $\textsc{Jump}_{|b_i|,\frac{|b_i|}{2}-k-1}$ problem with the following corrections:

- from all nonzero answers, a value of $\left\lfloor \frac{n-|b_i|}{2} \right\rfloor$ needs to be subtracted;
- at the optimum of the subproblem, zero will be returned.

The latter correction, however, does not change the algorithm very much, because the algorithm from Theorem 4 doesn't actually query the optimum point. The line 19 from Fig. 1 collects the partial answers one by one: it sets the bits of a_i at the corresponding positions from b_i to the previous partial answer ω_{i-1} and returns the updated value.

The complexity of the algorithm can be expressed as (here $n = qs + r$, $(0 < r \le s)$):

$$O(\sqrt{n}) + O\left(\frac{n}{\sqrt{s}}\right) + q(2+o_s(1))\frac{s}{\log_2 s} + (2+o_r(1))\frac{r}{\log_2 r} = \frac{(2+o_s(1))n}{\log_2 s}.$$

However, due to choice of s, it holds that $\log_2 s = (2+o_k(1))\log_2 k$, which finally results in $\frac{(1+o_{n-2\ell}(1))n}{\log_2(n-2\ell)}$. □

2.4 Upper Bound for Extreme Jump

The algorithm from Theorem 5 cannot be applied to the case of extreme JUMP, because $\left\lfloor \frac{n}{2} \right\rfloor - 1 - k = 0$. In this case we have to use another algorithm, which will be given in the proof of the following theorem.

Theorem 6. *The unrestricted black-box complexity of the extreme* JUMP *is at most* $n + \Theta(\sqrt{n})$.

Proof. As described in previous theorems, one can find a point x, such that $f(x) = \left\lfloor \frac{n}{2} \right\rfloor$, in $\Theta(\sqrt{n})$ queries. After that, if one flips two bits, the value of f remains the same iff one of these bits was correct and the other was not.

Once x is found, the algorithm tests $f(x \oplus 10^{i-1}10^{n-i-1})$ for all $i \in [2; n]$, and if it equals $\left\lfloor \frac{n}{2} \right\rfloor$, the value of b_i is set to zero, otherwise to one. This results in $n-1$ queries. After that, if the first bit is correct, then $0b_2 \dots b_n$ is the answer, otherwise its inverse is the answer. One has to make a single query to $f(0b_2 \dots b_n)$ to find which one is true. The complexity of this algorithm is $n + \Theta(\sqrt{n})$. □

3 The Matrix Lower Bound Theorem

In this section we present a new theorem which is similar to Theorem 2 from [4] except that the nodes corresponding to queries are required to be split in several types.

Theorem 7. *Let S be the search space of an optimization problem, and for each $s \in S$ there exists an instance such that s is a unique optimum. Let each query has one of T types, such that for any query q of the i-th type the following holds:*

- *there is exactly one answer to the query q which means that q is an optimum;*
- *there are at most $A_{i,j}$ answers such that the next query after such answer belongs to the j-th type.*

Define $B_{i,j}$, $1 \le i, j \le T + 2$, to be a matrix such that:

- $B_{i,j} = A_{j,i}$ *for* $1 \le i, j \le T$ *(note the transposition);*
- $B_{T+1,j} = 1$ *for* $1 \le j \le T + 1$*;*
- $B_{T+2,j} = 1$ *for* $1 \le j \le T + 2$*;*
- $B_{i,j} = 0$ *otherwise.*

Let the first ever query in the optimization process be of type 1. Define $V(d) = B^d \cdot (1, 0, \dots, 0)^T$ be a vector, $C(d) = V(d)_{T+1}$, $S(d) = V(d)_{T+2}$. Then the following statements are true:

1. *$C(d)$ is the maximum total number of possible queries with depth in $[1; d]$, where depth of a root is equal to one.*
2. *The lower bound on average depth of N nodes is $d + 1 - \frac{S(d)}{N}$ where d is an integer such that $C(d) \le N \le C(d + 1)$.*
3. *The unrestricted black-box complexity of the considered optimization problem is not less than the lower bound on average depth of $|S|$ nodes.*

Proof. According to Yao's minimax principle [6], the expected runtime of a randomized algorithm on any input is not less than the average runtime of the best deterministic algorithm over all possible inputs. Thus we construct a lower bound on complexity of a randomized algorithm by constructing a lower bound on the average performance of any deterministic algorithm over all possible inputs. A deterministic algorithm can be represented as a (rooted) decision tree with nodes corresponding to queries and arcs going downwards corresponding to answers to these queries. A total lower bound on the average performance of deterministic algorithms, just as in [4], is done by assigning $|S|$ different queries to different nodes of a tree such that their average depth is minimized, and then by considering all such trees and taking a minimum over them.

It should be noted that, if a (fixed) set of queries is to be assigned to nodes of a (fixed) rooted tree such that the average depth of these queries is minimized, an optimal assignment can be constructed in a greedy way: each query should be assigned to a free node with the minimum possible depth. Assume that an optimal assignment does not use at least one node a with depth d while using at least one node b with depth $d' > d$. Then one can move a query from the node b to the node a, which makes the average depth decrease, so the initial assignment is, in fact, not optimal.

Next we show that, in order to minimize the average depth, one needs to consider only the complete tree, that is, a tree where for any query of the i-th type, for any j there are exactly $A_{i,j}$ answers, each leading to a query of the j-th type. Indeed, if an optimal assignment can be done for an incomplete tree, it can be done for the complete tree as well, because all the nodes of any incomplete tree are preserved in the complete tree.

For a complete tree with the constraints determined by the matrix A (as specified in the theorem's statement) and with the root vertex of type 1, the number of vertices of type i and depth d (the root has the depth equal to 1) is exactly $\left(\left(A^T\right)^{d-1} \cdot (1, 0, \ldots, 0)^T\right)_i$. In the matrix B, the next-to-last row is designed to collect the sum of all numbers of vertices at all previous depths (which is exactly how $C(d)$ is defined), and the last row, in a similar manner, collects $S(d)$ — the sum of $C(i)$'s for all $1 \le i \le d$. In a more explicit way, $S(d)$ can be expressed as:

$$S(d) = \sum_{i=1}^{d} C(i) \cdot (d + 1 - i),$$

so the expression $C(d) \cdot (d+1) - S(d)$ is actually the sum of depths of all vertices up to the depth d:

$$C(d)(d + 1) - S(d) = \sum_{i=1}^{d} C(i) \cdot i,$$

and the expression $d + 1 - \frac{S(d)}{C(d)}$ is thus exactly the average depth of all such vertices.

If we consider arbitrary integer N, we can find an integer d such that $C(d) \le N \le C(d + 1)$. In this case, the total sum of depths of the first $C(d)$ vertices is

$C(d) \cdot (d+1) - S(d)$, and the next $N - C(d)$ vertices have the depth of $d+1$. The average depth is thus:

$$d_{avg}(N) = \frac{C(d) \cdot (d+1) - S(d) + (d+1) \cdot (N - C(d))}{N} = d + 1 - \frac{S(d)}{N}. \quad \square$$

It is difficult to use this theorem straightaway, because the lower bound on the average depth of N vertices is not defined only in terms of N and the matrix A, but additionally requires to find which depth d fulfils $C(d) \le N \le C(d+1)$. However, for several common usages it is possible to make it more convenient.

Theorem 8. *If there is only one type of queries in Theorem 7, and $A_{1,1} = k$ such that $k \ge 2$, then for the search space S the lower bound on the average depth is at least $\lfloor \log_k(1 + |S|(k-1)) \rfloor - \frac{1}{k-1}$.*

Proof. The value of $B^d \cdot (1,0,0)^T$ yields the following result (intermediate computations omitted):

$$\begin{pmatrix} k & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix}^d \cdot \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} k^d \\ \frac{k^d - 1}{k-1} \\ \frac{k^{d+1} - k - d(k-1)}{(k-1)^2} \end{pmatrix}.$$

One can see that $C(d) = \frac{k^d - 1}{k-1}$ and $S(d) = \frac{k^{d+1} - k - d(k-1)}{(k-1)^2}$.

Consider an equality $N = C(d) = \frac{k^d - 1}{k-1}$. It follows that:

$$d(N) = \log_k(1 + N(k-1)).$$

As for a given N we need to find an integer d such that $C(d) \le N < C(d+1)$, we need to round it down: $d = \lfloor d(N) \rfloor$.

Note that, if $d \ge 1$ and $k \ge 1$, $S(d)$ grows when d grows, as $S(d)' > 0$.

The expression for a lower bound on the average depth of N queries is at most:

$$\begin{aligned} d_{avg}(N) &= \lfloor d(N) \rfloor + 1 - \frac{S(\lfloor d(N) \rfloor)}{N} \ge \lfloor d(N) \rfloor + 1 - \frac{S(d(N))}{N} \\ &\ge \lfloor \log_k(1 + N(k-1)) \rfloor - \frac{1}{k-1}. \quad \square \end{aligned}$$

Note that the classical result from [4], the $\lfloor \log_{k+1} N \rfloor - 1$ lower bound, is actually not greater than the given bound. Indeed, for $k \ge 2$:

$$\begin{aligned} &\log_k(1 + N(k-1)) - \log_{k+1} N > \log_k(N(k-1)) - \log_{k+1} N \\ &= \log_k N - \log_{k+1} N + \log_k(k-1) > \log_k N - \log_{k+1} N > 0. \end{aligned}$$

For the case of $k = 1$, the lower bound is even stronger.

Theorem 9. *If there is only one type of queries in Theorem 7, and $A_{1,1} = 1$, then for the search space S the lower bound on the average depth is at least $(|S| + 1)/2$.*

Proof. In this case one can show that $C(d) = d$ and $S(d) = \frac{d^2 + d}{2}$. The average depth for N is $N + 1 - \frac{N^2 + N}{2N} = N + 1 - (N+1)/2 = (N+1)/2$. $\square$

4 Lower Bounds for $\textsc{Jump}_{n,\ell}$

First, let's apply Theorem 8 immediately to the JUMP problem.

Theorem 10. *For any n and $\ell < n/2$, the unrestricted black-box complexity of* $\textsc{Jump}_{n,\ell}$ *is at least* $\left\lfloor \log_{n-2\ell}(1 + 2^n(n - 2\ell - 1)) \right\rfloor - \frac{1}{n-2\ell-1}$.

Proof. In $\textsc{Jump}_{n,\ell}$, the search space has a size of 2^n. There are $n-2\ell+1$ possible answers to a query, but one of them terminates the search process immediately, so $k = n - 2\ell$. The result follows straightaway from Theorem 8.

Theorem 11. *The unrestricted black-box complexity of extreme* JUMP *for even n is at least $n - 1$.*

Proof. Follows from Theorem 10 by assuming $n - 2\ell = 2$.

The presented bounds are already an improvement over the currently known bounds (say, $\frac{n}{\log_2 3}$ for extreme JUMP and even n, as follows from [4]). However, for odd n Theorem 10 reports $\left\lfloor \log_3(1 + 2^{n+1}) \right\rfloor - 1/2$, which is still quite far away from the best known algorithms. Fortunately, the JUMP problem possesses a particular property, which can be used to refine the lower bounds using Theorem 7 with *two types* of queries.

Theorem 12. *For* $\textsc{Jump}_{n,\ell}$*, define an answer to the query to be* non-trivial *if it is neither* 0 *nor n. After receiving the first non-trivial answer for every subsequent query it is possible to determine* a priori *the parity of any non-trivial answer.*

Proof. Consider the optimum and a query. We introduce the following values:

- q_{00}: number of positions with zeros in both the optimum and the query;
- q_{01}: number of positions with zeros in the optimum and ones in the query;
- q_{10}: number of positions with ones in the optimum and zeros in the query;
- q_{11}: number of positions with ones in both the optimum and the query.

The number of zeros in the optimum modulo 2, which is $q_{00} \oplus q_{01}$, is fixed. The number of ones in the query modulo 2 is $q_{01} \oplus q_{11}$, and the answer to the query modulo 2 is $q_{00} \oplus q_{11}$. The following equality holds:

$$(q_{01} \oplus q_{11}) \oplus (q_{00} \oplus q_{11}) = q_{00} \oplus q_{01},$$

which means that the parity of the non-trivial answer is uniquely determined by the parity of the number of ones in the query.

As a result, if an algorithm receives the first non-trivial answer, all subsequent queries will provably have fewer possible answers. □

Using Theorem 12, we can define two types of queries to use with Theorem 7, namely, the queries happened before and after a non-trivial answer.

Theorem 13. *The unrestricted black-box complexity of* $\textsc{Jump}_{n,\ell}$ *for odd* n *is at least:*

$$\left\lfloor \log_{\frac{n-2\ell+1}{2}} \left(2^{n-2}(n-2\ell-1)+1\right)\right\rfloor - \frac{2}{n-2\ell-1}.$$

Proof. For odd n there are $n-2\ell+1 = 2k+2$ possible answers: one answer equal to 0, one answer equal to n and k pairs of non-trivial answers. For the Theorem 7, the first type of queries has $2k+1$ non-terminating answers, and the second type of queries, which occurs after one of $2k$ non-trivial answers is received from a query of the first type, has only $k+1$ non-terminating answers. The value of $B^d \cdot (1,0,0,0)^T$ is thus:

$$B^d \cdot \begin{pmatrix}1\\0\\0\\0\end{pmatrix} = \begin{pmatrix}1 & 0 & 0\,0\\ 2k & k+1 & 0\,0\\ 1 & 1 & 1\,0\\ 1 & 1 & 1\,1\end{pmatrix}^d \cdot \begin{pmatrix}1\\0\\0\\0\end{pmatrix} = \begin{pmatrix}1\\ 2\left((k+1)^d-1\right)\\ \frac{2(k+1)^d-dk-2}{k}\\ \frac{2(k+1)^{d+1}-\frac{(dk+2)^2+dk^2+4k}{2}}{k^2}\end{pmatrix}.$$

A problem of defining d in terms of N is more difficult this time: as $C(d) = \frac{2(k+1)^d-dk-2}{k}$, the equality $N = C(d)$ cannot be easily solved in terms of d. Instead, we introduce a function $d(N)$ such that the following equality holds:

$$N = \frac{2(k+1)^{d(N)} - d(N)k - 2}{k}.$$

We find the lower bound on the average depth $d_{avg}(N)$, keeping in mind that $S(d)$ grows as d grows and that $d(N) \geq 1$ for $N \geq 1$:

$$\begin{aligned}
d_{avg}(N) &= \lfloor d(N)\rfloor + 1 - \frac{S(\lfloor d(N)\rfloor)}{N} \geq \lfloor d(N)\rfloor + 1 - \frac{S(d(N))}{N}\\
&= \lfloor d(N)\rfloor + 1 - \frac{\frac{2(k+1)^{1+d(N)} - \frac{(d(N)k+2)^2+d(N)k^2+4k}{2}}{k^2}}{\frac{2(k+1)^{d(N)}-d(N)k-2}{k}}\\
&= \lfloor d(N)\rfloor + 1 - \frac{\frac{2(k+1)^{1+d(N)}-d(N)k^2-d(N)k-2k-2-\frac{d(N)k^2(d(N)-1)}{2}}{k}}{\frac{2(k+1)^{1+d(N)}-d(N)k^2-d(N)k-2k-2}{k+1}}\\
&\geq \lfloor d(N)\rfloor + 1 - \frac{k+1}{k} = \lfloor d(N)\rfloor - \frac{1}{k}.
\end{aligned}$$

We can also obtain a good lower bound on $d(N)$ by throwing out the $d(N)k$ part in the definition of $d(N)$ above, which leads to $d(N) > \log_{k+1}\left(\frac{Nk}{2}+1\right)$. Together, $d_{avg}(N) \geq \left\lfloor \log_{k+1}\left(\frac{Nk}{2}+1\right)\right\rfloor - \frac{1}{k}$. For $\textsc{Jump}_{n,\ell}$, it holds that $N = 2^n$ and $2k+2 = n-2\ell+1$, which constitutes:

$$\left\lfloor \log_{\frac{n-2\ell+1}{2}} \left(2^{n-2}(n-2\ell-1)+1\right)\right\rfloor - \frac{2}{n-2\ell-1}. \quad \square$$

Theorem 14. *The unrestricted black-box complexity of extreme* Jump *for odd* n *is at least* $n-2$.

Proof. For extreme JUMP and odd n, $n - 2\ell + 1 = 4$. Then from Theorem 13 it follows that the lower bound is at least:

$$\left\lfloor \log_2 \left(2^{n-2} \cdot 2 + 1\right) \right\rfloor - \frac{2}{2} = \left\lfloor \log_2 \left(2^{n-1} + 1\right) \right\rfloor - 1 \geq n - 2.$$

Theorem 15. *The unrestricted black-box complexity of* $\text{JUMP}_{n,\ell}$ *for even* n *is at least:*

$$\left\lfloor \log_{\frac{n-2\ell+2}{2}} \left(1 + 2^{n-1} \frac{(n-2\ell)^2}{n-2\ell-1}\right) \right\rfloor - \frac{2}{n-2\ell}.$$

Proof. For even n there are $n - 2\ell + 1 = 2k + 3$ possible answers ($k \geq 0$): one answer equal to 0, one answer equal to n, one answer equal to $n/2$ and k more pairs of non-trivial answers. For Theorem 7, the first type of queries has $2k + 2$ non-terminating answers, and the second type of queries can have either $k+1$ or k non-terminating answers, depending on the parity of the number of ones in a query. As we cannot predict the parity for all possible algorithms, the maximum number of queries is limited to $k + 1$. The matrix B has the following form:

$$B = \begin{pmatrix} 1 & 0 & 0\,0 \\ 2k+1 & k+2 & 0\,0 \\ 1 & 1 & 1\,0 \\ 1 & 1 & 1\,1 \end{pmatrix}.$$

We omit the intermediate computations and just state that:

$$C(d) = \frac{(2k+1)(k+2)^d - dk^2 - dk - 2k - 1}{(k+1)^2}$$

$$S(d) = \frac{(k+2)^d(2k^2+5k+2) - \frac{d^2k^3+dk^3+2d^2k^2+6dk^2+4k^2+2d+d^2k+7dk+10k+4}{2}}{(k+1)^3}.$$

Following the same approach as in the proof of Theorem 13, we define $d(N)$ such that $C(d(N)) = N$ and produce the following lower bound:

$$d_{avg}(N) \geq \lfloor d(N) \rfloor - \frac{1}{k+1}.$$

The lower bound on $d(N)$ can be achieved from the value of $C(d)$ by throwing out the $dk^2 + dk$ part, which yields:

$$d(N) \geq \log_{k+2} \left(1 + \frac{(k+1)^2 N}{2k+1}\right)$$

and, together:

$$d_{avg}(N) \geq \left\lfloor \log_{k+2} \left(1 + \frac{(k+1)^2 N}{2k+1}\right) \right\rfloor - \frac{1}{k+1}.$$

Substitution of N with 2^n and $2k + 2$ with $n - 2\ell$ proves the theorem. □

Note that Theorem 15 does not improve the bound for extreme JUMP and even n — it remains equal to $n-1$ when one sets $k = 0$ — because in this case the number of possible answers does not change after receiving the first non-trivial answer.

5 Conclusion

New black-box algorithms for solving $\textsc{Jump}_{n,\ell}$ problem are presented, giving the following upper bounds:

- for $\ell < n/2 - \sqrt{n}\log_2 n$: $\frac{2n(1+o(1))}{\log_2 n}$, where $o(1)$ is measured when $n \to \infty$;
- for $n/2 - \sqrt{n}\log_2 n \le \ell < \left\lfloor \frac{n}{2} \right\rfloor - 1$: $\frac{n(1+o(1))}{\log_2(n-2\ell)}$, where $o(1)$ is measured when $n - 2\ell \to \infty$;
- for $\ell = \left\lfloor \frac{n}{2} \right\rfloor - 1$: $n + \Theta(\sqrt{n})$.

A new theorem for constructing lower bounds on unrestricted black-box complexity of problems is proposed. The underlying idea is that influence of particular answers to queries to all subsequent queries can be formalized by assigning a type to each query and writing the relations in a form of a matrix. Several following steps for constructing the lower bounds are automated and can be performed using tools like Wolfram Alpha. We hope that this theorem can be used to obtain better lower bounds in other problems.

Using the proposed theorem, the lower bounds for $\textsc{Jump}_{n,\ell}$ are updated:

- for even n: $\left\lfloor \log_{\frac{n-2\ell+2}{2}} \left(1 + 2^{n-1}\frac{(n-2\ell)^2}{n-2\ell-1}\right) \right\rfloor - \frac{2}{n-2\ell} \ge \frac{n}{\log_2 \frac{n-2\ell+2}{2}} - 1$;
- for odd n: $\left\lfloor \log_{\frac{n-2\ell+1}{2}} \left(1 + 2^{n-2}(n-2\ell-1)\right) \right\rfloor - \frac{2}{n-2\ell-1} \ge \frac{n-1}{\log_2 \frac{n-2\ell+1}{2}} - 1$.

In particular, for extreme Jump the lower bounds become equal to $n-1$ for even n and $n-2$ for odd n. This means that the quotients at the first term of lower and upper bounds coincide. In the case of large, but not extreme ℓ, these quotients seem to coincide as well, however, the $(1+o(1))$ multiple can hide as much as $\left(\log_2(n-2\ell)/\log_2 \frac{n-2\ell+1}{2}\right)$, which makes it hard to see exactly.

This work was partially financially supported by the Government of Russian Federation, Grant 074-U01.

References

1. Doerr, B., Doerr, C., Ebel, F.: Lessons from the black-box: fast crossover-based genetic algorithms. In: Proceedings of Genetic and Evolutionary Computation Conference. pp. 781–788 (2014)
2. Doerr, B., Doerr, C., Kötzing, T.: Unbiased black-box complexities of jump functions, `http://arxiv.org/abs/1403.7806v2`
3. Doerr, B., Johannsen, D., Kötzing, T., Lehre, P.K., Wagner, M., Winzen, C.: Faster black-box algorithms through higher arity operators. In: Proceedings of Foundations of Genetic Algorithms. pp. 163–172 (2011)
4. Droste, S., Jansen, T., Wegener, I.: Upper and lower bounds for randomized search heuristics in black-box optimization. Theory of Computing Systems 39(4), 525–544 (2006)
5. Hoeffding, W.: Probability inequalities for sums of bounded random variables. Journal of the American Statistical Association 58(301), 13–30 (1963)
6. Yao, A.C.C.: Probabilistic computations: Toward a unified measure of complexity. In: Foundations of Computer Science, 1977., 18th Annual Symposium on. pp. 222–227 (1977)