Electronic Colloquium on Computational Complexity, Report No. 174 (2018)

Parity Decision Tree Complexity is Greater Than Granularity

Anastasiya Chistopolskaya* Vladimir V. Podolskii†

Abstract

We prove a new lower bound on the parity decision tree complexity $\mathsf{D}_\oplus(f)$ of a Boolean function f. Namely, granularity of the Boolean function f is the smallest k such that all Fourier coefficients of f are integer multiples of $1/2^k$. We show that $\mathsf{D}_\oplus(f) \geq k+1$.

This lower bound is an improvement of the known lower bound through the sparsity of f. Using our lower bound we determine the exact parity decision tree complexity of several important Boolean functions including majority, recursive majority and MOD^3 function. For majority the complexity is $n - \mathsf{B}(n) + 1$, where $\mathsf{B}(n)$ is the number of ones in the binary representation of n. For recursive majority the complexity is $\frac{n+1}{2}$. For MOD^3 the complexity is $n-1$ for n divisible by 3 and is n otherwise. Finally, we provide an example of a function for which our lower bound is not tight.

1 Introduction

Parity decision trees is a computational model in which we compute a known Boolean function $f\colon \{0,1\}^n \to \{-1,1\}$ on an unknown input $x \in \{0,1\}^n$ and in one query we can check the parity of arbitrary subset of inputs. The computational cost in this model is the number of queries we have made. The model is a natural generalization of a well-known decision trees model (in which only the value of a variable can be asked in one query) [1, 5].

Apart from being natural and interesting on its own parity decision trees model was studied mainly in connection with Communication Complexity and more specifically, with Log-rank Conjecture. In Communication Complexity most standard model there are two players Alice and Bob. Alice is given $x \in \{0,1\}^n$ and Bob is given $y \in \{0,1\}^n$ and they are trying to compute some fixed function $F\colon \{0,1\}^n \times \{0,1\}^n \to \{-1,1\}$ on input (x,y). The question is how many communication is needed to compute $F(x,y)$ in the worst case. It is known that the deterministic communication complexity $\mathsf{D}^{\mathsf{cc}}(F)$ of the function F is lower bounded by $\log \operatorname{rank}(M_F)$, where M_F is a communication matrix of F [6]. It is a long standing conjecture and one of the key open problems in Communication Complexity, called Log-rank Conjecture [7], to prove that $\mathsf{D}^{\mathsf{cc}}(F)$ is upper bounded by a polynomial of $\log \operatorname{rank}(M_F)$.

An important special case of Log-rank Conjecture addresses the case of XOR-functions $F(x,y) = f(x \oplus y)$ for some f, where $x \oplus y$ is a bit-wise XOR of Boolean vectors x and y. On one hand, this class of functions is wide and captures many important functions

*National Research University Higher School of Economics
†Steklov Mathematical Institute, Russian Academy of Sciences and National Research University Higher School of Economics

ISSN 1433-8092

(including equality, inner product, Hamming distance), and on the other hand the structure of XOR-functions allows to use analytic tools. For such functions $\mathrm{rank}(M_F)$ is equal to the Fourier sparsity $\mathsf{spar} f$, the number of non-zero Fourier coefficients of f. Thus, the Log-rank Conjecture for XOR-functions can be restated: is it true that $\mathsf{D}^{\mathsf{cc}}(F)$ is bounded by a polynomial of $\log \mathsf{spar} f$?

Given a XOR-function $f(x \oplus y)$ a natural way for Alice and Bob to compute the value of the function is to use a parity decision tree for f. They can simulate each query in the tree by computing parity of bits in their parts of the input separately and sending the results to each other. One query requires two bits of communication and thus $\mathsf{D}^{\mathsf{cc}}(F) \leq 2\mathsf{D}_{\oplus}(f)$. This leads to an approach to establish Log-rank Conjecture for XOR-function [16]: show that $\mathsf{D}_{\oplus}(f)$ is bounded by a polynomial of $\log \mathsf{spar} f$.

This approach received a lot of attention in recent years and drew attention to parity decision trees themselves [16, 13, 12, 15, 14, 4]. In a recent paper [4] it was shown that actually $\mathsf{D}^{\mathsf{cc}}(F)$ and $\mathsf{D}_{\oplus}(f)$ are polynomially related. This means that the simple protocol described above is not far from being optimal and that the parity decision tree version of Log-rank Conjecture stated above is actually equivalent to the original Log-rank Conjecture for XOR-functions.

All this motivates further research on parity decision trees. As for the lower bounds for parity decision tree complexity, one technique follows from the discussion above: $\mathsf{D}_{\oplus}(f) \geq \mathsf{D}^{\mathsf{cc}}(F)/2 \geq (\log \mathsf{spar} f)/2$. Although, if Log-rank conjecture for XOR-functions is true, this approach gives optimal bounds up to a polynomial, in many cases it does not help to determine precise parity decision tree complexity of Boolean functions. For example, this approach always gives bounds of at most $n/2$ for functions of n variables.

Another known approach is of a more combinatorial flavor. For standard decision trees there are several combinatorial measures known that lower bound decision tree complexity. Among them the most common are certificate complexity and block sensitivity. In [16] these measures were generalized to the setting of parity decision tree complexity. Parity decision tree complexity versions of these measures are actually known to be polynomially related to parity decision tree complexity [16]. However, they also do not give tight lower bounds for many interesting functions.

Examples of well-known functions for which the precise parity decision trees complexity is unknown include the majority function (playing a crucial role in many areas of Theoretical Computer Science, including Fourier analysis of Boolean functions), recursive majority (interesting, in particular, from decision tree complexity point of view as it provides a gap for deterministic and randomized decision tree complexity [11, 8]), MOD^3 function (that usually turns out to be hard for models limited to computation of parities).

Our Results In this paper we address the problem of improving known lower bounds for parity decision tree complexity. Our main result is a new lower bound in terms of the *granularity* of a Boolean function.

Granularity $\mathsf{gran}(f)$ of $f\colon \{0,1\}^n \to \{-1,1\}$ is the smallest k such that all Fourier coefficients of f are integer multiples of $1/2^k$. We show that

$$\mathsf{D}_{\oplus}(f) \geq \mathsf{gran}(f) + 1.$$

It is a simple corollary of Parseval's Identity that $\mathsf{gran}(f) \geq (\log \mathsf{spar}(f))/2$. Thus our lower bound is an improvement over the bound through sparsity. On the other hand, it was shown in [2] that $\mathsf{gran}(f) \leq \mathsf{spar}(f)$. Thus, this is an improvement by at most a factor of 2.

Despite for our lower bound being close to the lower bound through sparsity, it allows to prove tight lower bounds for several important functions. Also unlike the lower bound through sparsity, new approach allows to prove lower bounds up to n (the largest possible parity decision tree complexity of a function).

We hope that the connection between parity decision tree complexity and granularity will help to shed more light on the parity decision tree complexity.

We apply our lower bound to study the parity decision tree complexity of several well-known Boolean functions. We start with the majority function MAJ. We show that $\mathsf{D}_{\oplus}(MAJ) = n - \mathsf{B}(n) + 1$, where n is the number of variables and $\mathsf{B}(n)$ is the number of ones in the binary representation of n. The upper bound in this result is a simple adaptation of a folklore algorithm for the following problem (see, e.g. [10]). Suppose that for odd n we are given n balls of red and blue colors and we do not see the colors of the balls. In one query for any pair of balls we can check whether their colors are the same. Our goal is to find a ball of the same color as the majority of balls. We want to minimize the number of queries asked in the worst case. There is a folklore algorithm to solve this task in $n - \mathsf{B}(n)$ queries. It was shown in [10] that this is in fact optimal. On the idea level our lower bound for parity decision tree complexity is inspired by the proof of [10].

Next we proceed to recursive majority that computes an iteration of majority of three variables. We show that the parity decision tree complexity of this function is $(n+1)/2$. We also consider MOD^3 function that checks whether the number of ones in the input is divisible by 3. We show that the parity decision tree complexity of this function is $n-1$ for n divisible by 3 and is n for other values of n.

Finally, we show a series of examples of functions, for which our lower bound is not optimal. Namely, we consider threshold functions THR_n^l that check whether there are at least l ones in the input. We show that for $n = 8k+2$ for $k > 0$ and $l = 3$ our lower bound implies that at least $n-2$ queries are needed to compute the function, whereas the actual parity decision tree complexity is $n-1$. To prove this gap we combine our lower bound with an additional inductive argument allowing for a weak form of hardness amplification for the parity decision tree complexity of THR_n^k functions.

The rest of the paper is organized as follows. In Section 2 we provide necessary definition and preliminary information. In Section 3 we prove the lower bound on parity decision tree complexity. In Sections 4, 5 and 6 we study the parity decision tree complexity of majority, recursive majority and MOD^3 function respectively. Finally, in Section 7 we provide an example of a function for which our lower bound is not tight. Some of the technical proofs are moved to Appendix.

2 Preliminaries

2.1 Fourier Analysis

Throughout the paper we assume that Boolean functions are functions of the form $f\colon \{0,1\}^n \to \{-1,1\}$. That is, input bits are treated as 0 and 1 and to them we will usually apply operations over $\mathbb{F}_2$. Output bits are treated as -1 and 1 and the arithmetic will be over $\mathbb{R}$.

We denote the variables of functions by $x = (x_1, \ldots, x_n)$. We use the notation $[n] = \{1, \ldots, n\}$.

We briefly review the notation and needed facts from Boolean Fourier analysis. For extensive introduction see [9].

For functions $f, g\colon \{0,1\}^n \to \mathbb{R}$ consider an inner product

$$\langle f, g \rangle = \mathop{\mathbf{E}}_{x} f(x)g(x),$$

where the expectation is taken over uniform distribution of x on $\{0,1\}^n$.

For a subset $S \subseteq [n]$ we denote by $\chi_S(x) = \prod_{i \in S}(-1)^{x_i}$ the Fourier character corresponding to S. We denote by $\widehat{f}(S) = \langle f, \chi_S \rangle$ the corresponding Fourier coefficient of f.

It is well-known that for any $x \in \{0,1\}^n$ we have $f(x) = \sum_{S \subseteq [n]} \widehat{f}(S)\chi_S(x)$.

If $f\colon \{0,1\}^n \to \{-1,1\}$ (that is, if f is Boolean) then the well-known Parseval's Identity holds:

$$\sum_{S \subset [n]} \widehat{f}^2(S) = 1.$$

By the *support* of the Boolean function f we denote

$$\mathsf{Supp}(f) = \{S \subseteq [n] \mid \widehat{f}(S) \neq 0\}.$$

The *sparsity* of f is $\mathsf{spar}(f) = |\mathsf{Supp}(f)|$. Basically, the sparsity of f is the l_0-norm of the vector of its Fourier coefficients.

Consider a binary fraction α, that is α is a rational number that can be written in a form that its denominator is a power of 2. By the *granularity* $\mathsf{gran}(\alpha)$ of α we denote the minimal integer $k \geq 0$ such that $\alpha \cdot 2^k$ is an integer.

We will also frequently use the following closely related notation. For an integer L denote by $\mathsf{P}(L)$ the maximal power of 2 that divides L. It is convenient to set $\mathsf{P}(0) = \infty$.

Note that for Boolean f the Fourier coefficients of f are binary fractions. By the *granularity* of f we call the following value

$$\mathsf{gran}(f) = \max_{S \subseteq [n]} \mathsf{gran}(\widehat{f}(S)).$$

It is easy to see that for any $f\colon \{0,1\}^n \to \{-1,1\}$ it is true that

$$0 \leq \mathsf{gran}(f) \leq n - 1$$

and both of these bounds are achievable (for example, for $f(x) = \bigoplus_i x_i$ and $f(x) = \bigwedge_i x_i$ respectively).

It is known that $\mathsf{gran}(f)$ is always not far from the logarithm of $\mathsf{spar}(f)$:

$$\frac{\log \mathsf{spar}(f)}{2} \leq \mathsf{gran}(f) \leq \log \mathsf{spar}(f).$$

The first inequality can be easily obtained from Parseval's identity. The second is a non-trivial result [2, Theorem 3.3 for $\mu = 0$]. Again, both inequalities are tight (the first one is tight for inner product $\mathrm{IP}(x, y) = \bigoplus_i (x_i \wedge y_i)$ or any other bent function [9]; the second one is tight for example for $f(x) = \bigoplus_i x_i$).

2.2 Parity Decision Trees

A parity decision tree T is a rooted directed binary tree. Each of its leaves is labeled by -1 or 1, each internal vertex v is labeled by a parity function $\bigoplus_{i \in S_v} x_i$ for some subset $S_v \subseteq [n]$. Each internal node has two outgoing edges, one labeled by -1 and another by 1. A computation of T on input $x \in \{0, 1\}^n$ is the path from the root to one of the leaves that in each of the internal vertices v follows the edge, that has label equal to the value of $\bigoplus_{i \in S_v} x_i$. Label of the leaf that is reached by the path is the output of the computation. The tree T *computes* the function $f\colon \{0, 1\}^n \to \{-1, 1\}$ iff on each input $x \in \{0, 1\}^n$ the output of T is equal to $f(x)$. Parity decision tree complexity of f is the minimal depth of a tree computing f. We denote this value by $\mathsf{D}_\oplus(f)$.

One known way to lower bound parity decision tree complexity goes through communication complexity of XOR functions. We state the bound in the following lemma (see, e.g. [4]).

Lemma 1. *For any function $f\colon \{0, 1\}^n \to \{-1, 1\}$ we have*

$$\mathsf{D}_\oplus(f) \geq \frac{\log \mathsf{spar}(f)}{2}.$$

This lower bound turns out to be useful in many cases, especially when we are interested in the complexity up to a multiplicative constant or up to a polynomial factor. However, it does not always help to find an exact value of the complexity of the function and in principle cannot give lower bounds greater than $n/2$.

Another more combinatorial approach goes through analogs of certificate complexity and block sensitivity for parity decision trees [16]. Since parity block sensitivity is always less or equal then parity certificate complexity and we are interested in lower bounds, we will introduce only certificate complexity here.

For a function $f\colon \{0, 1\}^n \to \{-1, 1\}$ and $x \in \{0, 1\}^n$ denote by $C_\oplus(f, x)$ the minimal co-dimension of an affine subspace in $\{0, 1\}^n$ that contains x and on which f is constant. The *parity certificate complexity* of f is $C_\oplus(f) = \max_x C_\oplus(f, x)$.

Lemma 2 ([16])**.** *For any function $f\colon \{0, 1\}^n \to \{-1, 1\}$ we have*

$$\mathsf{D}_\oplus(f) \geq C_\oplus(f).$$

This approach allows to show strong lower bounds for some functions. For example, it can be used to show that $\mathsf{D}_\oplus(\mathrm{AND}_n) = n$. However, for more complicated functions like majority or recursive majority this lemma does not give tight lower bounds.

3 Lower Bound on Parity Decision Trees

Through the connection to communication complexity it is known that $\mathsf{D}_{\oplus}(f) \geq \frac{\log \mathsf{spar}(f)}{2}$ for any f. In our main result we improve this bound.

Theorem 3. *For any non-constant $f\colon \{0,1\}^n \to \{-1,1\}$ we have*

$$\mathsf{D}_{\oplus}(f) \geq \mathsf{gran}(f) + 1.$$

Proof. We prove the theorem by an adversary argument. That is, we will describe the strategy for the adversary to answer queries of a parity decision tree in order to make the tree to make many queries to compute the output.

Denote $k = \mathsf{gran}(f)$ and denote by $S \subseteq [n]$ the subset on which the granularity is achieved, that is $k = \mathsf{gran}(\widehat{f}(S))$. We have that

$$\begin{aligned}\widehat{f}(S) &= \frac{1}{2^n} \sum_{x\in\{0,1\}^n} f(x)\chi_S(x) = \frac{1}{2^n}\left(\sum_{x\in f^{-1}(1)} \chi_S(x) - \sum_{x\in f^{-1}(-1)} \chi_S(x)\right)\\ &= \frac{1}{2^n}\left(\sum_{x\in\{0,1\}^n} \chi_S(x) - 2\cdot \sum_{x\in f^{-1}(-1)} \chi_S(x)\right).\end{aligned}$$

Note that the first sum in the last expression is equal to 2^n if $S = \emptyset$ and is equal to 0 otherwise. Thus for the granularity of $\widehat{f}(S)$ to be equal to k the sum $\sum_{x\in f^{-1}(-1)} \chi_S(x)$ should be divisible by 2^{n-k-1} and should not be divisible by 2^{n-k}. In other words (recall that $\mathsf{P}(L)$ is the maximal power of 2 that divides L),

$$\mathsf{P}\left(\sum_{x\in f^{-1}(-1)} \chi_S(x)\right) = n-k-1. \tag{1}$$

After each step of the computation the query fixes some parity of inputs to be equal to some fixed value. Denote by $C_i \subseteq [n]$ the set of inputs that are still consistent with the current node of a tree after step i, and on which the function is equal to -1. We have that $C_0 = f^{-1}(-1)$.

We will show that we can answer the queries in such a way that

$$\mathsf{P}\left(\sum_{x\in C_{i+1}} \chi_S(x)\right) \leq \mathsf{P}\left(\sum_{x\in C_i} \chi_S(x)\right). \tag{2}$$

To see this observe that the $(i+1)$-st query splits the current set C_i into two disjoint subsets A and B. In particular,

$$\sum_{x\in C_i} \chi_S(x) = \sum_{x\in A} \chi_S(x) + \sum_{x\in B} \chi_S(x).$$

If both sums in the right-hand side are divisible by some power of 2, then the left-hand side also is. Thus,

$$\min\left(\mathsf{P}\left(\sum_{x\in A}\chi_S(x)\right),\mathsf{P}\left(\sum_{x\in B}\chi_S(x)\right)\right)\le \mathsf{P}\left(\sum_{x\in C_i}\chi_S(x)\right).$$

Pick for C_{i+1} the set, on which the minimum in the left-hand side is achieved.

Suppose the protocol makes t queries. The set of inputs that reach the leaf forms an affine subspace of Boolean cube of dimension at least $n-t$, on which the function f must be constant. Thus the sum

$$\sum_{x\in C_t}\chi_S(x)$$

is the sum of a character over an affine subspace, and thus is equal to either 0, or 2^{n-t}. In both cases

$$\mathsf{P}\left(\sum_{x\in A}\chi_S(x)\right)\ge n-t. \tag{3}$$

Combining (1)-(3) we get

$$n-k-1\ge n-t$$

and the theorem follows. □

4 Majority Function

In this section we analyze parity decision tree complexity of the majority function $\mathrm{MAJ}_n\colon\{0,1\}^n\to\{-1,1\}$. The function is defined as follows:

$$\mathrm{MAJ}_n(x)=-1\Leftrightarrow\sum_{i=1}^n x_i\ge\frac{n}{2}.$$

To state our results we will need the following notation: let $\mathsf{B}(k)$ be the number of ones in a binary representation of k.

We start with an upper bound. The following lemma is a simple adaptation of the folklore algorithm (see, e.g. [10]).

Lemma 4.

$$\mathsf{D}_{\oplus}(\mathrm{MAJ}_n)\le n-\mathsf{B}(n)+1.$$

Proof. Our parity decision tree will mostly make queries of the form $y\oplus z$ for a pair of variables. Note that such a query basically checks whether y and z are equal.

Our algorithm will maintain splitting of input variables into blocks of two types. We will maintain the following properties:

- the size of each block is a power of 2;
- all variables in each block of type 1 are equal;

- blocks of type 2 are balanced, that is they have equal number of ones and zeros.

In the beginning of the computation each variable forms a separate block of size one. During each step the algorithm will merge two blocks into a new one. Thus, after k steps the number of blocks is $n-k$.

The algorithms works as follows. On each step we pick two blocks of type 1 of equal size. We pick one variable from each block and query the parity of these two variables. If the variables are equal, we merge the blocks into a new block of type 1. If the variables are not equal, the new block is of type 2. The process stops when there are no blocks of type 1 of equal size.

It is easy to see that all of the properties listed above are maintained. In the end of the process we have some blocks of the second type (possibly none of them) and some blocks of the first type (possibly none of them) of pairwise non-equal size. Note that the value of the majority function is determined by the value of variables in the largest block of type 1. Indeed, all blocks of type 2 are balanced and the largest block of type 1 has more variables then all other blocks of type 1 in total. Thus, to find the value of MAJ_n it remains to query one variable from the largest block of type 1. Note, that the case when there are no blocks of type 1 in the end of the process correspond to balanced input (and even n). In this case we can tell that the output is -1 without any additional queries.

Note that the sum of sizes of all blocks is equal to n. Since the size of each block is a power of 2, there are at least $\mathsf{B}(n)$ blocks in the end of the computation (one cannot break n in the sum of less then $\mathsf{B}(n)$ powers of 2). Thus, overall we make at most $n-\mathsf{B}(n)+1$ queries and the lemma follows. $\square$

Before proceeding with the lower bound we briefly discuss lower bounds that can be obtained by other approaches. It is known that $\mathsf{spar}(\mathrm{MAJ}_n) = 2^{n-1}$ [9]. Thus from the sparsity lower bound we can only get $\mathsf{D}_{\oplus}(\mathrm{MAJ}_n) \geq \log \mathsf{spar}(\mathrm{MAJ}_n)/2 = \frac{n-1}{2}$.

Note also that each input $x \in \{0,1\}^n$ to MAJ_n lies in the subcube of dimension at least $\lceil \frac{n-1}{2} \rceil$. Indeed, if $\mathrm{MAJ}_n(x) = 1$ just pick a subcube on some subset of variables of size $\lceil \frac{n-1}{2} \rceil$ containing all ones of the input. The case $\mathrm{MAJ}_n(x) = -1$ is symmetrical. Thus, in the approach through certificate complexity we get $\mathsf{D}_{\oplus}(\mathrm{MAJ}_n) \geq \lceil \frac{n-1}{2} \rceil$.

We next show that Theorem 3 gives a tight lower bound for parity decision tree of MAJ_n.

Lemma 5. $\mathsf{gran}(\mathrm{MAJ}_n) = n - \mathsf{B}(n)$.

Proof. We will show that $\mathsf{gran}(\mathrm{MAJ}_n) \geq n - \mathsf{B}(n)$. The inequality in the other direction follows from Lemma 4.

We consider the Fourier coefficient $\widehat{\mathrm{MAJ}}_n([n])$ and show that its granularity is at least $n - \mathsf{B}(n)$. Let $k = \lfloor (n+1)/2 \rfloor$. Note that k is the smallest number such that MAJ_n is -1 on inputs with k ones.

Then we have

$$\begin{aligned}\widehat{\mathrm{MAJ}}_n([n]) &= \frac{1}{2^n}\left(\sum_{i=0}^{k-1}(-1)^i\binom{n}{i} - \sum_{i=k}^{n}(-1)^i\binom{n}{i}\right) \\ &= \frac{1}{2^n}\left(\sum_{i=0}^{n}(-1)^i\binom{n}{i} - 2\sum_{i=k}^{n}(-1)^i\binom{n}{i}\right) = \frac{1}{2^n}\left(0 - 2\sum_{i=k}^{n}(-1)^i\binom{n}{i}\right).\end{aligned}$$

From this we can see that

$$\mathsf{gran}(\widehat{\mathrm{MAJ}}_n([n])) = n - \mathsf{P}\left(2\sum_{i=k}^{n}(-1)^i\binom{n}{i}\right).$$

We proceed to simplify the sum of binomials (a very similar analysis is presented in [10]):

$$\sum_{i=k}^{n}(-1)^i\binom{n}{i} = \sum_{i=k}^{n}(-1)^i\left(\binom{n-1}{i-1}+\binom{n-1}{i}\right) = (-1)^k\binom{n-1}{k-1}.$$

Thus it remains to compute $\mathsf{P}(2\binom{n-1}{k-1})$. For even $n = 2h$ we have $k = h$ and $2\binom{n-1}{k-1} = 2\binom{2h-1}{h-1} = \binom{2h}{h}$. For odd $n = 2h+1$ we have $k = h+1$ and $2\binom{n-1}{k-1} = 2\binom{2h}{h}$.

By [10, Proposition 3.4] we have $\mathsf{P}(\binom{2h}{h}) = \mathsf{B}(h)$ (alternatively this can be seen from Kummer's theorem). Finally, notice that $\mathsf{B}(2h) = \mathsf{B}(h)$ and $\mathsf{B}(2h+1) = \mathsf{B}(h)+1$. It follows that

$$\mathsf{P}\left(2\sum_{i=k}^{n}(-1)^i\binom{n}{i}\right) = \mathsf{B}(n)$$

and

$$\mathsf{gran}(\widehat{\mathrm{MAJ}}_n([n])) = n - \mathsf{B}(n).$$

□

Overall, we have the following theorem.

Theorem 6.

$$\mathsf{D}_{\oplus}(\mathrm{MAJ}_n) = n - \mathsf{B}(n) + 1.$$

5 Recursive Majority

Next we study the parity decision tree complexity of recursive majority $\mathrm{MAJ}_3^{\otimes k}$. This is a function on $n = 3^k$ variables and it can be defined recursively. For $k = 1$ we just let $\mathrm{MAJ}_3^{\otimes 1} = \mathrm{MAJ}_3$. For $k > 1$ we let

$$\mathrm{MAJ}_3^{\otimes k} = \mathrm{MAJ}_3\left(\mathrm{MAJ}_3^{\otimes k-1}, \mathrm{MAJ}_3^{\otimes k-1}, \mathrm{MAJ}_3^{\otimes k-1}\right),$$

where each $\mathrm{MAJ}_3^{\otimes k-1}$ is applied to a separate block of variables.

We start with an upper bound.

Lemma 7. $\mathsf{D}_{\oplus}(\mathrm{MAJ}_3^{\otimes k}) \leq (n+1)/2$.

Proof. Basically, recursive majority $\mathrm{MAJ}_3^{\otimes k}$ is a function computed by a Boolean circuit which graph is a complete ternary tree of depth k, each internal vertex is labeled by the function MAJ_3 and each leaf is labeled by a (fresh) variable.

To construct an algorithm we first generalize the problem. We consider functions computed by Boolean circuits which graphs are ternary tree, where each non-leaf has fan-in 3 and is labeled by MAJ_3, and each leaf is labeled by a fresh variable. We will show that if the number of non-leaf variables in the circuit is l, then the function can be computed by a parity decision tree of size $l+1$.

The proof is by induction on l. If $l=1$, then the function in question is just MAJ_3 and by the results of Section 4 it can be computed by a parity decision tree of size 2.

For the step of induction consider a tree with l non-leaf vertices. Consider a non-leaf vertex of the largest depth. All of its three inputs must be variables, lets denote them by y, z and t, and in this vertex the function $\mathrm{MAJ}_3(y,z,t)$ is computed. Our first query will be $y \oplus z$. It will tell us whether y and z are equal. If $y=z$ are equal, then $\mathrm{MAJ}_3(y,z,t)=y$, and if $y \neq z$, then $\mathrm{MAJ}_3(y,z,t)=t$. Thus, we can substitute the gate in our vertex by the corresponding variable and reduce the problem to the circuit with $l-1$ non-leaf vertices. By induction hypothesis, the function computed by this circuit can be computed by at most $(l-1)+1=l$ queries. Thus, our original function is computable by $l+1$ queries.

It is left to observe that a complete ternary tree of depth k has $3^{k-1}+\ldots+1=\frac{3^k-1}{2}$ non-leaf vertices and for this tree our algorithm makes $\frac{3^k+1}{2}=\frac{n+1}{2}$ queries. $\square$

Before proceeding to the lower bound we again discuss lower bounds that can be obtained by other techniques.

First note that each input $x \in \{0,1\}^n$ lies in the subspace of co-dimension at most 2^k on which the function is constant. For this it is enough to show that in each x we can flip 3^k-2^k variables without changing the value of the function. This is easy to check by induction on k. For $k=1$ there are two variables that are equal to each other and we can flip the third variable without changing the value of the function. For $k>1$ consider inputs to the MAJ_3 at the top of the circuit. Two of them are equal and by induction hypothesis we can flip $3^{k-1}-2^{k-1}$ variables in each of them without changing the value of the function. The last input to the top gate does not affect the value of the function and we can flip all 3^{k-1} variables in it. Overall this gives us 3^k-2^k variables. This gives us $\mathsf{C}_{\oplus}(\mathrm{MAJ}_3^{\otimes k}) \leq 2^k = n^{\log_3 2}$ which does not give a matching lower bound.

For Fourier analytic considerations it is convenient to switch to $\{-1,1\}$ Boolean inputs. For a variable $y \in \{0,1\}$ let us denote by $y' \in \{-1,1\}$ the variable $y'=1-2y$. For now we will use new variables as inputs to Boolean functions.

The Fourier decomposition of MAJ_3 is

$$\mathrm{MAJ}_3(y',z',t') = \frac{1}{2}\left(y'+z'+t'-y'z't'\right). \tag{4}$$

From this the Fourier decomposition of $\mathrm{MAJ}_3^{\otimes k}$ can be obtain by recursion:

$$\begin{aligned}\mathrm{MAJ}_3^{\otimes k}(x^1,x^2,x^3) =& \frac{1}{2}(\mathrm{MAJ}_3^{\otimes k-1}(x^1)+\mathrm{MAJ}_3^{\otimes k-1}(x^2)+\mathrm{MAJ}_3^{\otimes k-1}(x^3)\\ &-\mathrm{MAJ}_3^{\otimes k-1}(x^1)\cdot\mathrm{MAJ}_3^{\otimes k-1}(x^2)\cdot\mathrm{MAJ}_3^{\otimes k-1}(x^3)),\end{aligned} \tag{5}$$

where x^1,x^2,x^3 are blocks of 3^{k-1} variables.

Lemma 1 can give lower bounds up to $n/2$ and thus in principle might give at least almost matching lower bound. However, this is not the case as we discuss below.

Note that since there is no free coefficient in the polynomial (4), Fourier coefficients arising from all three summands in the right-hand side of (5) will not cancel out with each other: none two of them have equal set of variables. Thus, if we denote $S(k) = \mathsf{spar}(\mathrm{MAJ}_3^{\otimes k})$ we have that $S(1) = 4$ and

$$S(k) = 3S(k-1) + S(k-1)^3 \tag{6}$$

for $k > 1$. On one hand, this means that $S(k) > S(k-1)^3$. This gives $S(k) > 2^{2\cdot 3^{k-1}}$. Thus $\log \mathsf{spar}(\mathrm{MAJ}_3^{\otimes k}) > 2 \cdot 3^{k-1} = 2n/3$ and $\mathsf{D}_{\oplus}(\mathrm{MAJ}_3^{\otimes k}) > n/3$.

On the other hand if we let $S'(k) = S(k) + 1/2$, it is easy to check that (6) implies

$$S'(k) < S'(k-1)^3.$$

Since $S'(1) = 9/2$ this gives $S'(k) < 2^{(\log_2 \frac{9}{2})\cdot 3^{k-1}}$. Thus,

$$\log \mathsf{spar}(\mathrm{MAJ}_3^{\otimes k}) < \left(\log_2 \frac{9}{2}\right) \cdot \frac{n}{3} < 0.723 \cdot n.$$

Thus Lemma 1 can give us a lower bound of at most $0.362 \cdot n$. We note that this upper bound on the sparsity can be further improved by letting $S'(k) = S(k) + \alpha$ for smaller α.

Now we proceed to the tight lower bound. Again we will estimate $\mathsf{gran}(\widehat{\mathrm{MAJ}_3^{\otimes k}}[n])$. Observe that this Fourier coefficient can be easily computed from (4) and (5). Indeed, from (4) we have that $\left|\widehat{\mathrm{MAJ}_3^{\otimes 1}}[n]\right| = \frac{1}{2}$. From (5) we have that

$$\left|\widehat{\mathrm{MAJ}_3^{\otimes k}}[n]\right| = \left|\frac{1}{2}(\widehat{\mathrm{MAJ}_3^{\otimes k-1}}[n])^3\right|.$$

The numerator of this Fourier coefficient equals to 1 for any k. Thus, denoting $G(n) = \mathsf{gran}(\widehat{\mathrm{MAJ}_3^{\otimes k}}[n])$ for $n = 3^k$ we have $G(3) = 1$ and

$$G(n) = 3G\left(\frac{n}{3}\right) + 1.$$

It is straightforward to check that $G(n) = \frac{n-1}{2}$. From this, Theorem 3 and Lemma 7 the following theorem follows.

Theorem 8. $\mathsf{D}_{\oplus}(\mathrm{MAJ}_3^{\otimes k}) = \frac{n+1}{2}$, *where $n = 3^k$ is the number of variables.*

6 MOD^3 Function

In this section we provide one more example of the well-known function for which our lower bounds allows to determine parity decision tree complexity.

We let

$$\mathrm{MOD}_n^3(x) = -1 \Leftrightarrow \sum_{i=1}^{n} x_i \equiv 0 \pmod 3,$$

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

For this function lower bounds through certificate complexity and sparsity are again not very strong.

For certificate complexity note that any $x \in \{0,1\}^n$ lies in the subspace of dimension approximately $n/3$ on which MOD^3_n is constant. Indeed, we can split x into $n/3 + o(n)$ blocks of size 3 in such a way that in each block all bits of x are equal. Flipping all variables in a block does not affect the value of the function. Thus, through certificate complexity we cannot obtain a lower bound better than $2n/3 + o(n)$.

For sparsity, again, Lemma 1 cannot give a lower bound better than $n/2$.

From Theorem 3, however, a much better lower bound follows easily.

Lemma 9. *For $n \equiv 0 \pmod 3$ we have $\mathsf{D}_{\oplus}(\mathrm{MOD}^3_n) \geq n-1$ and for $n \not\equiv 0 \pmod 3$ we have $\mathsf{D}_{\oplus}(\mathrm{MOD}^3_n) \geq n$.*

The proof of this lemma goes by analysis of $\widehat{\mathrm{MOD}^3_n}(\emptyset)$ and is technical. It can be found in Appendix 8.1.

We next show that this lower bound is tight. For this it is enough to show that $\mathsf{D}_{\oplus}(\mathrm{MOD}^3_n) \leq n-1$ for $n \equiv 0 \pmod 3$. We prove this by induction on n.

For $n = 3$ we just need to check whether all inputs x_1, x_2, x_3 are equal. It is enough to query $x_1 \oplus x_2$ and $x_2 \oplus x_3$. The output is -1 iff the output to both queries is 1.

For the step of induction consider the function $\mathrm{MOD}^3_{n+3}(x_1, \ldots, x_n)$. As in the proof of the upper bound for MAJ_n it is convenient to think of input variables as of trivial blocks of variables of size 1. By the first three queries we consecutively check whether $x_n = x_{n+1}$, $x_{n+2} = x_{n+3}$ and $x_n = x_{n+2}$ thus forming blocks of variables $\{x_n, x_{n+1}\}$, $\{x_{n+2}, x_{n+3}\}$ and $\{x_n, x_{n+1}, x_{n+2}, x_{n+3}\}$. If the answer to one of the queries is 'no', we know that the corresponding block is balanced. In this case we stop the process immediately (for example, we do not ask the second and the third queries above if the first block is balanced) and just query one variable from each of the remaining blocks. From this we know the number of ones in the input and can output $\mathrm{MOD}^3_{n+3}(x)$. It is easy to see that we make at most $n+2$ queries (we save one query by not querying variables from the balanced block). If on the other hand, the answer to all of the three first queries is 'yes', we know that the last four inputs are equal. Thus

$$\mathrm{MOD}^3_{n+3}(x_1, \ldots, x_{n+3}) = \mathrm{MOD}^3_n(x_1, \ldots, x_n)$$

and it remains to compute the latter function. By induction hypothesis this can be done in at most $n-1$ query and in total we again have at most $n+2$ queries.

Overall we get the following theorem.

Theorem 10. *For $n \equiv 0 \pmod 3$ we have $\mathsf{D}_{\oplus}(\mathrm{MOD}^3_n) = n-1$ and for $n \not\equiv 0 \pmod 3$ we have $\mathsf{D}_{\oplus}(\mathrm{MOD}^3_n) = n$.*

Remark 11. *We note that the functions MOD^k for larger k are trickier to analyze. For example, for MOD^4_6 both Fourier coefficients corresponding to $\emptyset$ and $[n]$ has granularity 1 giving lower bound $\mathsf{D}_{\oplus}(\mathrm{MOD}^4_6) \geq 2$. We need to consider a Fourier coefficient corresponding to a set of size 1 to show lower bound $\mathsf{D}_{\oplus}(\mathrm{MOD}^4_6) \geq 4$. This lower bound is tight: we can first by three queries form blocks of size 2, then pick one variable from each unbalanced block and compute the parity of them.*

7 A Function f with $\mathsf{D}_\oplus(f) > \mathsf{gran}(f) + 1$

In this section we provide an example of a function for which our lower bound is not tight. For this we study the family of threshold functions.

For arbitrary n and k we let

$$\mathrm{THR}_n^k(x) = -1 \Leftrightarrow \sum_{i=1}^{n} x_i \geq k,$$

where $x \in \{0,1\}^n$. Note that $\mathrm{MAJ}_n = \mathrm{THR}_n^{\lceil n/2 \rceil}$.

Our examples will form a subfamily of this family of functions.

To show that our lower bound is not tight we need an approach to prove even better lower bounds. We will do it via the following theorem.

Theorem 12. *For any s, k, n if $\mathsf{D}_\oplus(\mathrm{THR}_n^k) \geq s$, then $\mathsf{D}_\oplus(\mathrm{THR}_{n+2}^{k+1}) \geq s+1$.*

Proof. We will argue by a contradiction. Assume that $\mathsf{D}_\oplus(\mathrm{THR}_{n+2}^{k+1}) \leq s$. We will construct a parity decision tree for THR_n^k making no more than $s-1$ queries.

Denote the input variables to THR_n^k by $x = (x_1, \ldots, x_n)$. We introduce one more variable y (which we will fix later) and consider the sequence $x_1, \ldots, x_n, y, \neg y$ as inputs to the algorithm for THR_{n+2}^{k+1}. Note that $\mathrm{THR}_n^k(x) = \mathrm{THR}_{n+2}^{k+1}(x, y, \neg y)$. Our plan is to simulate the algorithm for THR_{n+2}^{k+1} on $(x, y, \neg y)$ and save one query on our way.

Consider the first query that the algorithm makes to $(x_1, \ldots, x_n, y, \neg y)$. Suppose first that the query does not ask the parity of all variables $(\bigoplus_{i=1}^n x_i) \oplus y \oplus \neg y$ (we will deal with this case later). Since the function THR_{n+2}^{k+1} is symmetric we can rename the input bits in such a way that the query contains input y and does not contain $\neg y$, that is the query asks the parity $(\bigoplus_{i \in S} x_i) \oplus y$ for some $S \subseteq [n]$. Now it is time for us to fix the value of y. We let $y = \bigoplus_{i \in S} x_i$. Then the answer to the first query is 0, we can skip it and proceed to the second query. For each next query of the algorithm for THR_{n+2}^{k+1} if it contains y or $\neg y$ (or both) we substitute them by $\bigoplus_{i \in S} x_i$ and $(\bigoplus_{i \in S} x_i) \oplus 1$ respectively. The result is the parity of some variables among $x_1, \ldots, x_n$ and we make this query to our original input x. Clearly the answer to the query to x is the same as the answer to the original query to $(x, y, \neg y)$. Thus, making at most $s-1$ queries we reach the leaf of the tree for THR_{n+2}^{k+1} and thus compute $\mathrm{THR}_{n+2}^{k+1}(x, y, \neg y) = \mathrm{THR}_n^k(x)$.

It remains to consider the case when the first query to THR_{n+2}^{k+1} is $(\bigoplus_{i=1}^n x_i) \oplus y \oplus \neg y$. This parity is equal to $\bigoplus_{i=1}^n x_i$ and we make this query to x. Now we proceed to the second query in the computation of THR_{n+2}^{k+1} and this query is not equal to $(\bigoplus_{i=1}^n x_i) \oplus y \oplus \neg y$. We perform the same analysis as above for this query: rename the inputs, fix y to the parity of subset of x to make the answer to the query to be equal to 0, simulate further queries to $(x, y, \neg y)$. Again we save one query in this case and compute $\mathrm{THR}_n^k(x)$ in at most $s-1$ queries. $\square$

Next we analyze the decision tree complexity of THR_n^2 functions. For them our lower bound is tight, but we need this analysis to use in combination with Theorem 12 to provide our example.

Lemma 13. *For even n we have $\mathsf{D}_{\oplus}(\mathrm{THR}_n^2) = n$ and for odd n we have $\mathsf{D}_{\oplus}(\mathrm{THR}_n^2) = n-1$.*

Proof sketch. The proof of the lower bound is technical and is omitted. The complete proof of the lemma can be found in Appendix 8.2.

Here we only prove that the lower bound is tight for odd n. To provide an algorithm making at most $n-1$ queries we again will split variables into blocks and again will assume that in the beginning all blocks are of size 1. We split all variables but one into pairs and check whether variables in each pair are equal. After this we have $(n-1)/2$ blocks of size 2 and one block of size 1. If there is a balanced block of size 2, again we can just query one variable from each of the remaining blocks thus learning the number of ones in the input. This allows us to compute the function in at most $n-1$ queries. If all blocks of size 2 contain equal variables, then note that the value of the function does not depend on the variable in the block of size 1. Indeed, $\mathrm{THR}_n^2(x) = 1$ iff $\sum_i x_i \geq 2$ iff there is a block of size 2 containing variables equal to 1. Thus it remains to query one varaible from each block of size 2, which again alows us to compute the function with at most $n-1$ queries. $\square$

We are now ready to proceed to the example of the functions for which the lower bound in Theorem 3 is tight.

Lemma 14. *For $n = 8k+2$ for integer k we have $\mathsf{gran}(\mathrm{THR}_n^3) = n-3$.*

The proof of this lemma is technical and can be found in Appendix 8.3.

We now show that for functions in Lemma 14 their decision tree complexity is greater than their granularity plus one.

Theorem 15. *For $n = 8k+2$ for integer $k > 0$ we have $\mathsf{D}_{\oplus}(\mathrm{THR}_n^3) = n-1$.*

Proof. For the lower bound we note that $n-3$ is odd and thus by Lemma 13 we have $\mathsf{D}_{\oplus}(\mathrm{THR}_{n-2}^2) \geq n-2$. Then by Theorem 12 we have $\mathsf{D}_{\oplus}(\mathrm{THR}_n^3) \geq n-1$.

For the upper bound we again view the inputs as blocks of size 1 and by checking equality of variables combine all variables but two into blocks of size 4. If we encounter a balanced block we just query one variable from all remaining block thus learning the number of ones in the input in at most $n-1$ queries. If all blocks contain equal variables, then as in the proof of Lemma 13 we observe that two variables outside of blocks of size 4 does not affect the value of the function. Indeed, $\mathrm{THR}_n^3(x) = 1$ iff $\sum_i x_i \geq 3$ iff there is a block of size 4 containing variables equal to 1. $\square$

Thus, we have shown that the lower bound in Theorem 3 is not tight for THR_{8k+2}^3. However, the gap between the lower bound and the actual complexity is 1.

Remark 16. *We note that from our analysis it is straightforward to determine the complexity of THR_n^3 for all n. If $n = 4k$ or $4k+3$ for some k, then $\mathsf{D}_{\oplus}(THR_n^3) = n$ and if $n = 4k+1$ or $n = 4k+2$, then $\mathsf{D}_{\oplus}(THR_n^3) = n-1$. The lower bounds (apart from the case covered by Theorem 15) follows from the consideration of $\widehat{\mathrm{THR}}_n^3(\emptyset)$ and $\widehat{\mathrm{THR}}_n^3([n])$ as in the proof of Lemma 14. The upper bound follows the same analysis as in the proof of Theorem 15.*

References

[1] H. Buhrman and R. de Wolf. Complexity measures and decision tree complexity: a survey. *Theor. Comput. Sci.*, 288(1):21–43, 2002.

[2] P. Gopalan, R. O'Donnell, R. A. Servedio, A. Shpilka, and K. Wimmer. Testing fourier dimensionality and sparsity. *SIAM J. Comput.*, 40(4):1075–1100, 2011.

[3] H. Gould. *Combinatorial identities: a standardized set of tables listing 500 binomial coefficient summations*. Morgantown, W Va, 1972.

[4] H. Hatami, K. Hosseini, and S. Lovett. Structure of protocols for XOR functions. In *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, pages 282–288, 2016.

[5] S. Jukna. *Boolean Function Complexity - Advances and Frontiers*, volume 27 of *Algorithms and combinatorics*. Springer, 2012.

[6] E. Kushilevitz and N. Nisan. *Communication complexity*. Cambridge University Press, 1997.

[7] L. Lovász and M. E. Saks. Lattices, Möbius functions and communication complexity. In *29th Annual Symposium on Foundations of Computer Science, White Plains, New York, USA, 24-26 October 1988*, pages 81–90, 1988.

[8] F. Magniez, A. Nayak, M. Santha, J. Sherman, G. Tardos, and D. Xiao. Improved bounds for the randomized decision tree complexity of recursive majority. *Random Struct. Algorithms*, 48(3):612–638, 2016.

[9] R. O'Donnell. *Analysis of Boolean Functions*. Cambridge University Press, 2014.

[10] M. E. Saks and M. Werman. On computing majority by comparisons. *Combinatorica*, 11(4):383–387, 1991.

[11] M. E. Saks and A. Wigderson. Probabilistic boolean decision trees and the complexity of evaluating game trees. In *27th Annual Symposium on Foundations of Computer Science, Toronto, Canada, 27-29 October 1986*, pages 29–38. IEEE Computer Society, 1986.

[12] A. Shpilka, A. Tal, and B. lee Volk. On the structure of boolean functions with small spectral norm. In *Innovations in Theoretical Computer Science, ITCS'14, Princeton, NJ, USA, January 12-14, 2014*, pages 37–48, 2014.

[13] H. Y. Tsang, C. H. Wong, N. Xie, and S. Zhang. Fourier sparsity, spectral norm, and the log-rank conjecture. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pages 658–667, 2013.

[14] H. Y. Tsang, N. Xie, and S. Zhang. Fourier sparsity of GF(2) polynomials. In *Computer Science - Theory and Applications - 11th International Computer Science Symposium in*

Russia, CSR 2016, St. Petersburg, Russia, June 9-13, 2016, Proceedings, pages 409–424, 2016.

[15] P. Yao. Parity decision tree complexity and 4-party communication complexity of xor-functions are polynomially equivalent. *Chicago J. Theor. Comput. Sci.*, 2016, 2016.

[16] Z. Zhang and Y. Shi. On the parity complexity measures of boolean functions. *Theor. Comput. Sci.*, 411(26-28):2612–2618, 2010.

8 Appendix: Omitted Proofs

8.1 Proof of Lemma 9

We consider the Fourier coefficient $\widehat{\mathrm{MOD}}_n^3(\emptyset)$ and compute its granularity.

We have

$$\widehat{\mathrm{MOD}}_n^3(\emptyset) = \frac{1}{2^n}\left(\sum_{i\not\equiv 0 \pmod 3}\binom{n}{i} - \sum_{i\equiv 0 \pmod 3}\binom{n}{i}\right)$$
$$= \frac{1}{2^n}\left(\sum_{i=0}^{n}\binom{n}{i} - 2\sum_{i\equiv 0 \pmod 3}\binom{n}{i}\right) = \frac{1}{2^n}\left(2^n - 2\sum_{i\equiv 0 \pmod 3}\binom{n}{i}\right).$$

From this we can see that $\mathsf{gran}(\widehat{\mathrm{MOD}}_n^3(\emptyset)) = n - \mathsf{P}\left(\sum_{i\equiv 0 \pmod 3}\binom{n}{i}\right) - 1$ and thus

$$\mathsf{D}_{\oplus}(\mathrm{MOD}_n^3) \geq n - \mathsf{P}\left(\sum_{i\equiv 0 \pmod 3}\binom{n}{i}\right). \tag{7}$$

It is well known (see, e.g. [3, equation 1.56]) that

$$\sum_{i\equiv 0 \pmod 3}\binom{n}{i} = \frac{2^n+m}{3},$$

where m is equal to 1 or -1 for $n \not\equiv 0 \pmod 3$ and m is equal to 2 or -2 for $n \equiv 0 \pmod 3$. It is easy to see that in the former case $\mathsf{P}\left(\sum_{i\equiv 0 \pmod 3}\binom{n}{i}\right) = 0$ and in the latter case $\mathsf{P}\left(\sum_{i\equiv 0 \pmod 3}\binom{n}{i}\right) = 1$. From this and (7) the lemma follows.

8.2 Complete Proof of Lemma 13

We start with a lower bound.

Here we will need to consider two Fourier coefficients, $\widehat{\mathrm{THR}}_n^2(\emptyset)$ and $\widehat{\mathrm{THR}}_n^2([n])$. We start with the latter one.

We have

$$\widehat{\mathrm{THR}}_n^2([n]) = \frac{1}{2^n}\left(\sum_{i=0}^{1}(-1)^i\binom{n}{i} - \sum_{i=2}^{n}(-1)^i\binom{n}{i}\right)$$
$$= \frac{1}{2^n}\left(2\sum_{i=0}^{1}(-1)^n\binom{n}{i} - \sum_{i=0}^{n}(-1)^n\binom{n}{i}\right) = \frac{1}{2^n}\left(2\sum_{i=0}^{1}(-1)^n\binom{n}{i} - 0\right).$$

From this we can see that $\mathsf{gran}(\widehat{\mathrm{THR}}_n^2([n])) = n - \mathsf{P}\left(\sum_{i=0}^{1}(-1)^n\binom{n}{i}\right) - 1$ and thus

$$\mathsf{D}_{\oplus}(\mathrm{THR}_n^2) \geq n - \mathsf{P}\left(\sum_{i=0}^{1}(-1)^n\binom{n}{i}\right).$$

By the same analysis for $\widehat{\mathrm{THR}}_n^2(\emptyset)$ we can show that

$$\mathsf{D}_{\oplus}(\mathrm{THR}_n^2) \geq n - \mathsf{P}\left(\sum_{i=0}^{1}\binom{n}{i}\right).$$

Note that $\sum_{i=0}^{1}(-1)^n\binom{n}{i} = 1-n$ and $\sum_{i=0}^{1}\binom{n}{i} = 1+n$. From this for even n we clearly obtain a lower bound of $\mathsf{D}_{\oplus}(\mathrm{THR}_n^2) \geq n$. For odd n it is easy to see that one of the numbers $1-n$ and $1+n$ is not divisible by 4. Thus for odd n we obtain lower bound $\mathsf{D}_{\oplus}(\mathrm{THR}_n^2) \geq n-1$.

It remains to prove that the lower bound is tight for odd n. To provide an algorithm making at most $n-1$ queries we again will split variables into blocks and again will assume that in the beginning all blocks are of size 1. We split all variables but one into pairs and check whether variables in each pair are equal. After this we have $(n-1)/2$ blocks of size 2 and one block of size 1. If there is a balanced block of size 2, again we can just query one variable from each of the remaining blocks thus learning the number of ones in the input. This allows us to compute the function in at most $n-1$ queries. If all blocks of size 2 contain equal variables, then note that the value of the function does not depend on the variable in the block of size 1. Indeed, $\mathrm{THR}_n^2(x) = 1$ iff $\sum_i x_i \geq 2$ iff there is a block of size 2 containing variables equal to 1. Thus it remains to query one varaible from each block of size 2, which again alows us to compute the function with at most $n-1$ queries.

8.3 Proof of Lemma 14

For the upper bound we need to consider an arbitrary Fourier coefficient $\widehat{\mathrm{THR}}_n^3(S)$. We have

$$\widehat{\mathrm{THR}}_n^3([S]) = \frac{1}{2^n}\left(\sum_{x,|x|\leq 2}\chi_S(x) - \sum_{x,|x|\geq 3}\chi_S(x)\right) = \frac{1}{2^n}\left(2\sum_{x,|x|\leq 2}\chi_S(x) - \sum_{x\in\{0,1\}^n}\chi_S(x)\right),$$

where by $|x|$ we denote $\sum_{i=1}^{n} x_i$. The second sum in the last expression is equal to either 2^n or 0 depending on S. Thus we have

$$\mathsf{gran}(\widehat{\mathrm{THR}}_n^3(S)) = n - \mathsf{P}\left(\sum_{x,|x|\leq 2}\chi_S(x)\right) - 1. \tag{8}$$

Denote the size of S by l. Then we have

$$\sum_{x,|x|\leq 2} \chi_S(x) = 1 - l + (n-l) + \frac{l(l-1)}{2} - l(n-l) + \frac{(n-l)(n-l-1)}{2},$$

where the first summand corresponds to x with $|x| = 0$, the next two summands correspond to $|x| = 1$ and the last three correspond to $|x| = 2$.

Rearranging this expression we obtain

$$\sum_{x,|x|\leq 2} \chi_S(x) = \frac{4l^2 + 2 + (n+1)(n-4l)}{2}.$$

We need to show that for $n \equiv 2 \pmod 8$ this number is divisible by 4, that is its numerator is divisible by 8. Since divisibility by 8 depends only on the remainder of n when divided by 8, it is enough to check divisibility of the numerator by 8 for $n = 2$. We have

$$4l^2 + 2 + (n+1)(n-4l) = 4l^2 + 2 + 3(2-4l) = 4(l^2 - 3l + 2),$$

which is clearly divisible by 8 for all l. Thus $\mathsf{P}\left(\sum_{x,|x|\leq 2} \chi_S(x)\right) \geq 2$ for $n = 8k+2$ and

$$\mathsf{gran}(\mathrm{THR}_n^3) \leq n - 3.$$

For the lower bound on the granularity it is enough to consider Fourier coefficients $\widehat{\mathrm{THR}}_n^3(\emptyset)$ and $\widehat{\mathrm{THR}}_n^3([n])$. For them we have

$$\sum_{x,|x|\leq 2} \chi_\emptyset(x) = 1 + n + \frac{n(n-1)}{2} = \frac{2 + n(n+1)}{2}$$

and

$$\sum_{x,|x|\leq 2} \chi_{[n]}(x) = 1 - n + \frac{n(n-1)}{2} = \frac{2 + n(n-3)}{2}.$$

To show the lower bound it is enough to show that for any $n = 8k+2$ at least one of these expressions is not divisible by 8, that is their numerators are not divisible by 16. It is straightforward to check that for $n \equiv 2 \pmod{16}$ we have $2 + n(n+1) \equiv 8 \pmod{16}$ and for $n \equiv 10 \pmod{16}$ we have $2 + n(n-3) \equiv 8 \pmod{16}$. In both cases by (8) we found a Fourier coefficients with granularity at least $n - 3$.

ECCC ISSN 1433-8092

https://eccc.weizmann.ac.il