UGP Report

Sankalp Mittal

April 2024

Contents

1 Introduction

This is a report on the UGP on Algebraic Complexity and a summary of what was learnt during the semester.

The "Holy Grail" of Complexity Theory is the $P \overset{?}{\neq} NP$ problem. This problem has been unexpectedly hard to solve.
Leslie Valiant hypothesised $VP \overset{?}{\neq} VNP$ problem as a stepping stone towards the $P \overset{?}{\neq} NP$ problem.
Recent works in Complexity Theory have been trying to show the separation between various classes that have sprung out of Valiant's initial classes.

In this report, I will cover some attempts at finding out lower bounds for some special cases.

2 Preliminaries

2.1 Basic Terminology

Firstly, we will introduce the basic definitions of a few terms which we will be using

- **Algebraic Circuit:** An *Algebraic Circuit* is formally a *Directed Acyclic Graph* with a unique *sink vertex* called the *root* such that each internal vertex is labelled as $+$ or $\times$.
- **Formula:** A formula is the same as a circuit with the constraint that any node present in the formula can have *at most one* outgoing edge.
- **Algebraic Branching Circuits:** An ABP is a *layered directed acyclic graph* with edges labelled with linear polynomials. There is a *source vertex (s)* and a *sink vertex (t)*.

2.2 Complexity Classes

- **Algebraic Circuits** are divided into two complexity classes **Valiant's P (VP)** and **Valiant's NP (VNP)**
  - **VP**: Consists of family of polynomials whose size and degree are both polynomially bounded.
  - **VNP**: Set of all families of polynomials $f \in \mathbb{F}[x_1, x_2, \cdots, x_n]$ such that there exists a polynomial in VP, $g \in \mathbb{F}[x_1, \cdots, x_n, y_1, \cdots, y_m]$, with $m = poly(n)$ and,

$$f(\mathbf{x}) = \sum_{\mathbf{a} \in \{0,1\}^m} g(\mathbf{x}, \mathbf{a})$$

- **Formulas** have defined on them class **VF**, it is the set of families of polynomials, $f \in \mathbb{F}[x_1, x_2, \cdots, x_n]$ that can be represented by a formula of size $\mathbf{poly(n)}$.
- **Arithmetic Branching Programs** have defined on them **VBP**, it is defined as the set of the families of polynomials having an ABP of size $\mathbf{poly(n)}$.

2.3 Restricted types of Circuits

2.3.1 Multilinear Circuits

A **multilinear polynomial** is one in which is *homogeneous*, each variable has *individual degree at-most* 1.

A **multilinear circuit** is one that computes a *mutilinear polynomial* from the subtree present at each of its nodes.

2.3.2 Set-Multilinear Circuits

Consider a partition of a set of variables $\{X_1, X_2 \cdots X_d\}$
A **set-multilinear polynomial** is one that is multilinear and and each monomial has *exactly one variable* from each of the d sets.

A **set-multilinear circuit** is one that computes a *set-mutilinear polynomial* with respect to a partition from the subtree present at each of its nodes.

2.4 Restricted types of ABPs

2.4.1 Set-Multilinear ABP (smABP)

Consider a partition of a set of variables $\{X_1, X_2 \cdots X_d\}$ A **set-multilinear ABP** is a $d+1$ layered ABP such that all labels on edges that go from i to $i+1$ have labels in a variable from X_i.

2.4.2 Read-once Oblivious ABP (ROABP)

An ABP is said to be **oblivious** if, for each layer, all the edge labels are **univariate** polynomials in a single variable.
An ABP is called a **read-once oblivious ABP (ROABP)** if each variable appears in at most one layer.

2.5 Restricted types of Formulas

2.5.1 Multilinear Formulas

An arithmetic formula is *multilinear* if the polynomial computed by each gate of the formula is *multilinear*.

2.5.2 Syntactic Multilinear formula

If in a formula, at each *multiplication node*(v), the subformula(Φ_v) is such that the set of variables in the subtree made at the two child nodes are disjoint, the formula is called *syntactic multilinear*.

3 Hardness Bootstrapping

There is a theorem that shows how lower bounds on $\sum smABP$ can lead to the separation of classes VBP and VNP.
Theorem 3.1: *Let n, d be integers such that $d = O(log\ n/log\ log\ n)$. Let $P_{n,d}$ be a set-multilinear polynomial in VNP of degree d. If $P_{n,d}$ cannot be computed by a $\sum smABP$ of width $poly(n)$, then $VBP \neq VNP$.* [2]

Lemma 3.1: *If a set-multilinear polynomial, with d sets each of size $\leq n$ can be calculated by a ABP of size s, it can be calculated by a sum of smABP's of total width $d^{O(d)}s$.* [2]

Using *Lemma 3.1*, we know that any ABP computing a *set-multilinear polynomial* can be replaced by a *similar sized* (in the *low degree* regime) smABP and then, we will proceed using proof by contradiction,
Using the assumption $P_{n,d}$ cannot be computed note that the width of the $\sum$smABP is, $n^{\omega(1)}$, and $d^{O(d)}s \geq n^{\omega(1)}$. Using the fact $d = O(\log n/\log\log n)$ we get $d^{O(d)} = poly(n)$, hence $s = n^{\omega(1)}$, and we have our desired seperation.

We need a few other lemmas to prove *Lemma 3.1*, these are as follows
Lemma 3.2: *A degree d polynomial f that can be computed by an ABP of size s can also be computed by an homogeneous ABP of width s and length d.* [2]
Proof: To homogenize an ABP we will

- Divide each vertex,v into d vertices such that each vertex $v^{(i)}$ computes the homogeneous part with degree i.
- Now, we will arrange all the vertex that compute the same degree in the same layer. This gives *width s* and length d, hence **Lemma 3.2**.
- This works because a vertex $v^{(i)}$ can only have edges going to $u^{(i)}$ or $u^{(i+1)}$ (each edge is labelled by a linear form).

Lemma 3.3: *If a homogenous ABP of width w and length d for a set-multilinear polynomial is needed, then the polynomial can be calculated by a $\sum smABP$ of width $d!w$.* [2]
Proof:
We begin by writing the ABP for a sm-polynomial in its IMM form,

$$P_{n,d} = \prod_{i}^{d} M_i,$$

Here each M is a matrix with $w \times w$ entries that are linear forms. Write each $M_i = \sum_{j=1}^{d} M_{ij}$, such that each M_{ij} has terms from the set X_j.

$$P_{n,d} = \prod_{i}^{d} \sum_{j=1}^{d} M_{ij}$$

We can ignore the non-set-multilinear terms, i.e. ignore the products having terms like $M_{ij}, M_{i'j}$. This gives the form

$$P_{n,d} = \sum_{\pi \in S_d} \prod_{i=1}^{d} M_{i\pi(i)},$$

Hence we have a sum of $d!$ smABP's each of width w

Hence we can easily prove *Lemma 1* using *Lemma 3.2* and *Lemma 3.3* back to back, using the fact

$$d! \approx d^{O(d)}$$

4 Presentable VNP

We will look at some definitions and techniques necessary to understand this section.

4.1 Approximate Circuits

There are a few polynomials that can be computed approximately in VP, if we just provide a little degree of freedom.
Instead of feeding polynomials in the inputs of the circuit we feed polynomials in ϵ and then calculate the polynomial $g \in \mathbb{F}[\epsilon][x_1, \cdots, x_n]$. We say $f \in \mathbb{F}[x_1, \cdots, x_n]$ is approximated by g to an order of approximation M if

$$g(\boldsymbol{x}, \epsilon) = \epsilon^M f(\boldsymbol{x}) + \epsilon^{M+1} Q(\boldsymbol{x}, \epsilon)$$

If the size of g over the field $\mathbb{F}[\epsilon]$ is in VP, then f is in $\overline{\text{VP}}$.
If we restrict the class such that the polynomials in ϵ used must have polynomial-sized circuits themselves, we get a new class called $\overline{\text{VP}}_\epsilon$. This is also called *presentable VP*.
Similar classes can be defined in the cases of VNP.

4.2 Interpolation

Consider a polynomial $P(x_1, \cdots, x_n, y)$ and $deg_y P = d$ then the polynomial can be written as,

$$P(x_1, \cdots, x_n, y) = P_0(x_1, \cdots, x_n) + P_1(x_1, \cdots, x_n)y + \cdots + P_d(x_1, \cdots, x_n)y^d$$

Consider d distinct constants $\alpha_0, \cdots, \alpha_n$ and let $P(\alpha_i) = P(x_1, \cdots, x_n, \alpha_i)$. Each of the P_i can be calculated using the following matrix multiplication.

$$\begin{bmatrix} P_0 \\ P_1 \\ \vdots \\ P_d \end{bmatrix} = \begin{bmatrix} 1 & \alpha_0 & \cdots & \alpha_0^d \\ 1 & \alpha_1 & \cdots & \alpha_1^d \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha_d & \cdots & \alpha_d^d \end{bmatrix}^{-1} \begin{bmatrix} P(\alpha_0) \\ P(\alpha_1) \\ \vdots \\ P(\alpha_d) \end{bmatrix}$$

This method is used to find out the coefficients of one of the variables in any polynomial.

4.3 Valiant's Criterion

Let $f = \sum_e c_e \boldsymbol{x}^{\boldsymbol{e}}$ be a polynomial in n variables of degree $poly(n)$ over a field $\mathbb{F}$. Suppose that there exists a string function $\phi : \{0,1\}^* \to \{0,1\}^*$ in #P/poly such that $\phi(\langle \boldsymbol{e} \rangle) = \langle c_{\boldsymbol{e}} \rangle$. Then, the polynomial f is in VNP over the field $\mathbb{F}$.

4.4 Results

Theorem 4.1: *Over any finite field* $VNP = \overline{VNP}_\epsilon$.[1]
Proof:
To prove this we will show $VNP \subseteq \overline{VNP}_\epsilon$ and $\overline{VNP}_\epsilon \subseteq VNP$.
The first containment is trivial from the definition. We will proceed as follows for the second one
We have access to $f \in \overline{\mathrm{VNP}}_\epsilon$ using the approximation

$$g(\boldsymbol{x}, \epsilon) = \epsilon^M f(\boldsymbol{x}) + \epsilon^{M+1} Q(\boldsymbol{x}, \epsilon)$$

This is of the hypercube form,

$$g(\boldsymbol{x}, \epsilon) = \sum_{\boldsymbol{a} \in \{0,1\}^m} h(\boldsymbol{x}, \boldsymbol{a}, \epsilon)$$

We will extract the coefficients of $\epsilon^M \boldsymbol{x}^{\boldsymbol{e}}$ in g using *interpolation* and taking the interpolation points to be the *roots of unity.* Consequently $c_{\boldsymbol{e}}$ can be obtained as a hypercube sum of an *exponential degree* circuit of *polynomial size.*

Using finite field arithmetic and the closure of the Boolean class #P under exponential sums, we can go to the *boolean world.*

Thus, we can show that the algebraic circuit above can be replaced by a (multi-output) *Boolean circuit* of polynomial size and the hypercube sum computing the coefficient function is demonstrated in *#P/poly.*

Now using Valiant's criterion we can claim,

$$\overline{VNP}_\epsilon \subseteq VNP \implies \overline{VNP}_\epsilon = VNP$$

5 Super-Polynomial lower bounds for Multi-Linear Formulas

We will be discussing the proof of the fact that *multi-linear formulas for permanent and determinant are of super-polynomial size*; for this purpose, we will be looking at some definitions and techniques necessary to understand this section.

5.1 Partial Derivative Matrix (or Coefficient Matrix)

Let f be a multilinear polynomial over the set of variables $\{y_1, \cdots, y_m\} \cup \{z_1, \cdots, z_m\}$. For a multilinear monomial p in the set of variables $\{y_1, \cdots, y_m\}$ and a multilinear monomial q in the set of variables $\{z_1, \cdots, z_m\}$, denote by $M_f(p, q)$ the coefficient of the monomial pq in the polynomial f. Since the number of multilinear monomials in a set of m variables is 2^m, we can think of M_f as a $2^m \times 2^m$ matrix, with entries in the field F.

For each node v, Φ_v represents the formula made by the subtree of v, and denote by Y_v the set of variables in $\{y_1, \cdots, y_m\}$ that appear in Φ_v and similarly Z_v is the set of variables in $\{z_1, \cdots, z_m\}$ that appear in Φ_v.

$$b(v) = \frac{|Y_v| + |Z_v|}{2}$$

$$a(v) = min(|Y_v| + |Z_v|)$$

$$d(v) = b(v) - a(v)$$

5.1.1 Rank of the Matrix

We are interested in the rank of the matrix M_v. The following is a set of results regarding the same.

1. $\textbf{Rank}(M_v) \leq 2^{a(v)}$
   This result is easy to see as this is bounding the rank by the number of *non zero* rows or columns, whichever is *smaller*.

2. $\textbf{Rank}(M_v) \leq \textbf{Rank}(M_{v_1}) + \textbf{Rank}(M_{v_2})$
   This is a result that basically states

$$\textbf{Rank}(A + B) \leq \textbf{Rank}(A) + \textbf{Rank}(B)$$

3. $\textbf{Rank}(M_v) = \textbf{Rank}(M_{v_1}) \cdot \textbf{Rank}(M_{v_2})$
   This result is a direct result of the tensor product of two matrices.

5.2 k-unbalanced

5.2.1 k-unbalanced nodes

We say that a node v is *k-unbalanced* if $d(v) \geq k$.

5.2.2 k-unbalanced paths

Let γ be a simple path from a leaf w to a node v of the formula. We say that γ is *k-unbalanced* if it contains at least one k-unbalanced node.

5.2.3 Central Paths

We say that γ is *central* if for every u, u_1 on the path γ , such that u_1 is a direct son of u (i.e., there is an edge from u_1 to u), we have $b(u) \leq 2b(u_1)$.

5.2.4 k-weak node

We say that a node v of the formula is *k-weak* if every central path that reaches v is k-unbalanced.

5.3 Results

Lemma 5.1: *Let Φ be a syntactic multilinear arithmetic formula over the set of variables $\{y_1, \cdots, y_m\} \cup \{z_1, \cdots, z_m\}$ and let v be a node in Φ. If v is k-weak then,* [4]

$$\mathbf{Rank}(M_v) \leq |\Phi_v| \cdot 2^{b(v)-k/2}$$

Proof: This can be proved by considering the base case of v being a leaf and then conditioning based on if v is a

- k-unbalanced node
- plus gate
- product gate

We are aiming to show that $|\Phi_v|$ is small, for our final proof for lower bounds.

5.3.1 Random Partition and Assignment

X is the set of variables present in the formula. We think of X as a matrix of variables with n rows and n columns. Let $m = \lceil n^{1/3} \rceil$.

First, choose, uniformly at random, for every $i \in [m]$ two values $q_i, r_i \in [n]$, such that all the $2m$ chosen values are different. (We think of q_i, r_i as the indices of rows). Then choose, uniformly at random, for every $i \in [m]$ two additional values $s_i, t_i \in [n]$, such that all the $2m$ chosen values are different. (We think of s_i, t_i as the indices of columns).
With equal probability, one of the following two assignments is done,

$$\begin{pmatrix} x_{q_i,s_i} & x_{q_i,t_i} \\ x_{r_i,s_i} & x_{r_i,t_i} \end{pmatrix} \longrightarrow \begin{pmatrix} y_i & z_i \\ 1 & 1 \end{pmatrix}$$

$$\begin{pmatrix} x_{q_i,s_i} & x_{q_i,t_i} \\ x_{r_i,s_i} & x_{r_i,t_i} \end{pmatrix} \longrightarrow \begin{pmatrix} y_i & 1 \\ z_i & 1 \end{pmatrix}$$

All other variables in X are assigned values in $\{0, 1\}$.

Lemma 5.2: *Let Φ be a syntactic multilinear arithmetic formula over the set of variables $X = \{x_{i,j}\}_{i,j\in[n]}$, such that every variable in X appears in Φ, and such that $|\Phi| \leq n^{\epsilon log\ n}$ where ϵ is a small enough universal constant (e.g., $\epsilon = 10^{-6}$). Let A be a random assignment to the variables in X, as above. Then, with probability of $1 - o(1)$ the formula A is **k-weak**, for $k = n^{1/32}$.*[4]

We show that with a very large probability, the formula after the assignment A is *k-weak* for $k = n^{1/32}$.

Proof: We will show that for every v_i, the probability that v_i is not k-unbalanced is smaller than $O(n^{-\delta})$, even when conditioning on the event that $v_1, \cdots, v_{i-1}$ are not k-unbalanced.

Here we require $k = n^{1/32}$ this is as during the construction of the proof, we require that $d(v_i) < k$ is not true and as $d(v_i)$ has to be an integer which has the probability of $O(k \cdot n^{-1/16})$, for this to be of the required order $O(n^{-1/32})$ we need $k = n^{1/32}$.

Theorem 5.3: *Any multilinear arithmetic formula for the permanent or the determinant of an $n \times n$ matrix (over any field) is of size $n^{\Omega(log\ n)}$.* [4]
Proof:
Permanent and determinant follow similar proofs, we use proof by contradiction, assume that $|\Phi| \leq n^{\epsilon \log n}$.
Let A be a random assignment to the variables in X, as defined earlier. Then, Φ_A is a syntactic multilinear arithmetic formula over the set of variables $\{y_1, \cdots, y_m\} \cup \{z_1, \cdots, z_m\}$. With probability of $1 - o(1)$ the formula Φ_A is k-weak, for $k = n^{1/32}$. Hence, with a probability of $1 - o(1)$.

$$\textbf{Rank}(M_{\Phi_A}) \leq n^{\epsilon \log n} \cdot 2^{m-k/2} < 2^m \tag{1}$$

At the other hand, since the output of Φ is the permanent of X, the output of Φ_A must be the permanent of the matrix $\{A(x_{i,j})\}_{i,j\in[n]}$. By the definition of A, the *permanent* of that matrix is always,

$$f(y_1, \cdots, y_m, z_1, \cdots, z_m) \equiv \prod_{i=1}^{m}(y_i + z_i)$$

And the *determinant* of that matrix is,

$$g(y_1, \cdots, y_m, z_1, \cdots, z_m) \equiv \prod_{i=1}^{m}(y_i - z_i)$$

Note that M_f is a permutation matrix, because for every multilinear monomial p in the set of variables $\{y_1, \cdots, y_m\}$ there is exactly one multilinear monomial q in the set of variables $\{z_1, \cdots, z_m\}$ such that $M_f(p,q) = 1$ (and otherwise $M_f(p,q) = 0$), and vice-versa. Hence,

$$\mathbf{Rank}(M_{\Phi_A}) = \mathbf{Rank}(M_f) = 2^m$$

This is in direct contradiction to 1, and hence our initial assumption was wrong, and the bounds are superpolynomial.

6 Superpolynomial Lower Bounds for low depth Circuits

The following basic definitions would be helpful in this section. This section is based on proving superpolynomial lower bounds for *constant-depth* arithmetic circuits.

6.1 Hardness Escalation

This is a technique used to prove lower bounds on more general circuits by proving lower bounds on restricted forms on arithmetic circuits.
This is widely used in proving results in many areas of computational complexity.

6.2 Polynomial Identity Testing (PIT)

PIT is the problem of efficiently determining whether two multivariate polynomials are identical. More formally, a PIT algorithm is given a circuit that computes p in a field and decides whether p is the *zero polynomial.* Finding *deterministic algorithms* for PIT is one of the most important open problems in algebraic computing complexity.

6.2.1 PIT and Lower Bounds

People have shown that superpolynomial lower bounds for general algebraic circuits imply deterministic sub-exponential time algorithms for general PIT.
It has also been shown that the hardness of constant-depth circuits implies deterministic PIT for constant-depth circuits.

6.3 Results

Theorem 6.1: *Let N, d, Δ be growing parameters with $d = o(log\, N)$. Assume $\mathbb{F}$ has characteristic 0 or greater than d. There is an explicit polynomial $P_{N,d}(x_1, \cdots, x_N)$ that has no algebraic circuits of product-depth Δ and size at most $N^{d^{exp(-O(\Delta))}}$.* [3]

Theorem 6.2: *Assume $d \leq (log\, n)/100$. For any product-depth $\Delta \geq 1$, any set-multilinear circuit C computing $IMM_{n,d}$ of product-depth at most Δ must have size at least $n^{d^{exp(-O(\Delta))}}$. In the particular case that $\Delta = 2$, the size of C must be at least $n^{\Omega(\sqrt{d})}$.* [3]

Any homogeneous circuit computing a set-multilinear polynomial can be converted into a set-multilinear circuit of the same depth and size $s{\cdot}d^{O(d)}$, combining with *Theorem 6.2* we get,

Corollary 6.3: *Assume $d \leq (log\, n)/100$. For any product-depth $\Delta \geq 1$, any* *homogeneous circuit C computing $IMM_{n,d}$ of product-depth at most Δ must have size at least $n^{d^{exp(-O(\Delta))}}$. In the particular case that $\Delta = 2$, the size of C must be at least $n^{\Omega(\sqrt{d})}$.* [3]

Any (possibly non-homogeneous) algebraic circuit of product depth Δ and size s computing a homogeneous polynomial P of degree d can be converted to a homogeneous circuit for P of product-depth 2Δ and size poly$(s) \cdot d^{O(d)}$. This conversion assumes that the underlying field has characteristic 0 or greater than d. Combining with *Theorem 6.2* we get,

Corollary 6.4: *Assume $d \leq (log\ n)/100$ and char$(\mathbb{F}) = 0$ or greater than d. For any product-depth $\Delta \geq 1$, any algebraic circuit C computing $IMM_{n,d}$ of product-depth at most Δ must have size at least $n^{d^{exp(-O(\Delta))}}$. In the particular case that $\Delta = 1$, the size of C must be at least $n^{\Omega(\sqrt{d})}$.* [3]

This will allow us to directly show **Theorem 6.1** as here $N = dn^2$ and this allows us to say log $n/100 = o(\log N)$.

Theorem 6.2 also allows us to prove the following theorem for constant-depth Algebraic circuits. Which means that circuits of depth Γ are *superpolynomially* more powerful than circuits of depth $\Gamma - 1$.

Theorem 6.5: *Assume that the underlying field $\mathbb{F}$ has characteristic 0. For any constant $\Gamma \geq 2$ and s a growing parameter, there exists a set-multilinear polynomial Q_Γ of depth Γ and size s such that any depth $(\Gamma - 1)$ circuit computing Q_Γ must have size $s^{\omega(1)}$.*[3]

Using the *PIT and Lower Bound* relation we can also derive the following result,

Corollary 6.6: *Let $\mu > 0$ be a real number and $\mathbb{F}$ a field of characteristic 0. Let C be an algebraic circuit of size $s \leq poly(n)$, depth $\Delta = o(log\ log\ log\ n)$ computing a polynomial on n variables, then there is a deterministic algorithm that can check whether the polynomial computed by C is identically zero or not in time $(s^{\Delta+1} \cdot n)^{O(n^\mu)}$.*[3]

Listing down the ideas for all these proofs is out of the scope of this report, so we will be listing down only the proof ideas for some of the results for the **special case of $\boldsymbol{\Delta = 2}$**.

6.4 Proofs for $\Delta = 2$

6.4.1 Relative Rank (relrk)

Consider the set of words on an alphabet $A \subseteq \mathbb{Z} \backslash \{0\}$. Let $w = (w_1, \cdots, w_d) \in A^d$. For $S \subseteq [d]$, let w_s denote $\sum_{i \in S} w_i$ and $\mathcal{P}_w = \{i | w_i \geq 0\}$ and $\mathcal{N}_w = \{i | w_i < 0\}$.

We say $w \in A^d$ is *b-unbiased* if $|w_{[d]}| \leq b$ for every $t \leq d$.
Let $\mathcal{M}_w^{\mathcal{P}}$ and $\mathcal{M}_w^{\mathcal{N}}$ denote the sets of set-multilinear monomials over only the positive and only the negative variable sets.
Define $M_w(f)$ as the matrix of size $|\mathcal{M}_w^{\mathcal{P}}| \times |\mathcal{M}_w^{\mathcal{F}}|$. Where the coefficient at the entry (m_1, m_2) is the coefficient of the monomial $m_1 m_2$ in f. Then define the **relative rank** as,

$$\text{relrk}_w(f) = \frac{\text{rank}(M_w(f))}{\sqrt{|\mathcal{M}_w^{\mathcal{P}}| \cdot |\mathcal{M}_w^{\mathcal{F}}|}}$$

The following properties are similar to the previous section

1. $\text{relrk}_w(f) \leq 2^{-|w_{[d]}|/2}$
2. $\text{relrk}_w(f+g) \leq \text{relrk}_w(f) + \text{relrk}_w(g)$
3. If $f = f_1 \cdot f_2 \cdot \dots \cdot f_t$, then

$$\text{relrk}_w(f) = \prod_{i \in [t]} \text{relrk}_w(f_i)$$

6.4.2 Proof of Theorem 6.2, Corollary 6.3 and Corollary 6.4

Lemma 6.7: *Let $w \in A^d$ be any word which is b-unbiased. If there is a set-multilinear circuit computing $IMM_{2^b,d}$ of size s and product-depth Δ then there is also a set-multilinear circuit of size s and product-depth Δ computing a polynomial $P_w \in \mathbb{F}_{sm}[\overline{X}(w)]$ such that $relrk_w(P_w) \geq 2^{-b/2}$.*[3]

Claim 6.8: *Let $d \geq 16$ and $k > 2\sqrt{d}$ be an integer. Let w be any word of length d on the alphabet $\{-k, \lfloor k - k/\sqrt{d} \rfloor\}$. Then, any set-multilinear formula C of product depth 2 and of size s satisfies.*[3]

$$\text{relrk}_w(C) \leq s \cdot 2^{-\frac{k\sqrt{d}}{8}}$$

This is proved by dividing C as $C = C_1 + \dots C_t$ where each C_i is of the form $\prod\sum\prod\sum$ and then we make cases based on if C_i has degree $\geq \sqrt{d}/2$.

Using the two above results, we can obtain the following results,
Fix $k = \lfloor \log_2 n \rfloor \implies k > 2\sqrt{d}$, We can construct by induction a word w on the alphabet which is k-unbiased, if $|w|_{[i]} \leq 0$, we choose $w_{i+1} = \lfloor k - k/\sqrt{d} \rfloor$, else choose $w_{i+1} = -k$.

$$s \geq 2^{\frac{k\sqrt{d}}{8}} 2^{-k} \geq 2^{(\log_2 n)\frac{\sqrt{d}}{8} - \log_2 n} \geq n^{\frac{\sqrt{d}}{8}} 2^{-\frac{\log_2 n}{16} - \log_2 n} \geq n^{\frac{\sqrt{d}}{8} - \frac{17}{16}}$$

Lemma 6.8: *For $n \geq 4^{\sqrt{d}+1}$, any set-multilinear circuit C of product-depth 2 computing $IMM_{n,d}$ has size at least $n^{\Omega(\sqrt{d})}$.*[3]
This directly proves **Theorem 6.2 and Corollary 6.3** for $\Delta = 2$ and **Corollary 6.4** for $\Delta = 1$ as the product depth doubles on *homogenisation.*

References

[1] C.S. Bhargav, Prateek Dwivedi, and Nitin Saxena. Learning the coefficients: A presentable version of border complexity and applications to circuit factoring. 2023.

[2] C.S. Bhargav, Prateek Dwivedi, and Nitin Saxena. Lower bounds for the sum of small-size algebraic branching programs. 2023.

[3] Nutan Limaye, Srikanth Srinivasan, and Sebastien Tavenas. Superpolynomial lower bounds against low-depth algebraic circuits. 2021.

[4] Ran Raz. Multi-linear formulas for permanent and determinant are of super-polynomial size. 2009.