Lower Bounds for some Algebraic Models of Computation

Prerona Chatterjee (Visiting Faculty Member, NISER Bhubaneswar)

November 11, 2024

Complexity Theory

Addition v.s. Multiplication v.s. Factorisation

Complexity Theory

Addition	v.s.	Multiplication	v.s.	Factorisation
Class I	v.s.	Class II	v.s.	Class IV/V

Addition	v.s.	Multiplication	v.s.	Factorisation
Class I	v.s.	Class II	v.s.	Class IV/V

- Why?

Complexity Theory

Addition	v.s.	Multiplication	v.s.	Factorisation
Class I	v.s.	Class II	v.s.	Class IV/V

- Why? Addition seems easier than Multiplication which seems easier than Factorisation.

Addition	v.s.	Multiplication	v.s.	Factorisation
Class I	v.s.	Class II	v.s.	Class IV/V

- Why? Addition seems easier than Multiplication which seems easier than Factorisation.
- Can one formalise this intuition?

Complexity Theory

Addition	v.s.	Multiplication	v.s.	Factorisation
Class I	v.s.	Class II	v.s.	Class IV/V

- Why? Addition seems easier than Multiplication which seems easier than Factorisation.
- Can one formalise this intuition? That is what Complexity Theory tries to do.

Complexity Theory

Addition	v.s.	Multiplication	v.s.	Factorisation
Class I	v.s.	Class II	v.s.	Class IV/V

- Why? Addition seems easier than Multiplication which seems easier than Factorisation.
- Can one formalise this intuition? That is what Complexity Theory tries to do.

Q: Given a computational problem and constraints on the computational power at hand,

Complexity Theory

Addition	v.s.	Multiplication	v.s.	Factorisation
Class I	v.s.	Class II	v.s.	Class IV/V

- Why? Addition seems easier than Multiplication which seems easier than Factorisation.
- Can one formalise this intuition? That is what Complexity Theory tries to do.

Q: Given a computational problem and constraints on the computational power at hand,

- design a computational model that captures the constraints

Complexity Theory

Addition	v.s.	Multiplication	v.s.	Factorisation
Class I	v.s.	Class II	v.s.	Class IV/V

- Why? Addition seems easier than Multiplication which seems easier than Factorisation.
- Can one formalise this intuition? That is what Complexity Theory tries to do.

Q: Given a computational problem and constraints on the computational power at hand,

- design a computational model that captures the constraints
- study the amount of resource required by the model to complete the task.

Complexity Theory: Major Sub-Areas

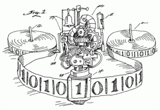
$P \stackrel{?}{=} NP$: Are easily verifiable boolean functions easy to compute as well?

Complexity Theory: Major Sub-Areas

$P \stackrel{?}{=} NP$: Are easily verifiable boolean functions easy to compute as well?

Traditional Time Complexity

Given a boolean function f on n inputs, how many steps are required by a Turing machine to compute the f (in terms of n)?

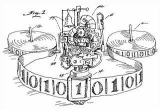


Complexity Theory: Major Sub-Areas

$\mathrm{P} \stackrel{?}{=} \mathrm{NP}$: Are easily verifiable boolean functions easy to compute as well?

Traditional Time Complexity

Given a boolean function f on n inputs, how many steps are required by a Turing machine to compute the f (in terms of n)?



Traditional Space Complexity

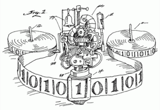
Given a boolean function f on n inputs, how much space is required by a Turing machine to compute the f (in terms of n)?

Complexity Theory: Major Sub-Areas

$P \stackrel{?}{=} NP$: Are easily verifiable boolean functions easy to compute as well?

Traditional Time Complexity

Given a boolean function f on n inputs, how many steps are required by a Turing machine to compute the f (in terms of n)?



Traditional Space Complexity

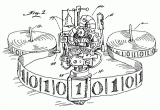
Given a boolean function f on n inputs, how much space is required by a Turing machine to compute the f (in terms of n)?

Circuit Complexity

Complexity Theory: Major Sub-Areas

$P \stackrel{?}{=} NP$: Are easily verifiable boolean functions easy to compute as well?

Traditional Time Complexity
Given a boolean function f on n inputs, how many steps are required by a Turing machine to compute the f (in terms of n)?



Traditional Space Complexity
Given a boolean function f on n inputs, how much space is required by a Turing machine to compute the f (in terms of n)?

Circuit Complexity

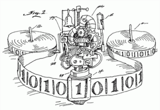
Communication Complexity

Complexity Theory: Major Sub-Areas

$P \stackrel{?}{=} NP$: Are easily verifiable boolean functions easy to compute as well?

Traditional Time Complexity

Given a boolean function f on n inputs, how many steps are required by a Turing machine to compute the f (in terms of n)?



Traditional Space Complexity

Given a boolean function f on n inputs, how much space is required by a Turing machine to compute the f (in terms of n)?

Circuit Complexity

Communication Complexity

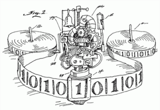
Proof Complexity

Complexity Theory: Major Sub-Areas

$P \stackrel{?}{=} NP$: Are easily verifiable boolean functions easy to compute as well?

Traditional Time Complexity

Given a boolean function f on n inputs, how many steps are required by a Turing machine to compute the f (in terms of n)?



Traditional Space Complexity

Given a boolean function f on n inputs, how much space is required by a Turing machine to compute the f (in terms of n)?

Circuit Complexity

Communication Complexity

Proof Complexity

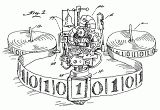
Quantum Computation

Complexity Theory: Major Sub-Areas

$P \stackrel{?}{=} NP$: Are easily verifiable boolean functions easy to compute as well?

Traditional Time Complexity

Given a boolean function f on n inputs, how many steps are required by a Turing machine to compute the f (in terms of n)?



Traditional Space Complexity

Given a boolean function f on n inputs, how much space is required by a Turing machine to compute the f (in terms of n)?

Circuit Complexity

Communication Complexity

Proof Complexity

Quantum Computation

Algebraic Computation

Algebraic Circuit Complexity

Q: What is the most succinct way of representing the given polynomial of interest?

Algebraic Circuit Complexity

Q: What is the most succinct way of representing the given polynomial of interest?

An n-variate, degree d-polynomial has $\binom{n+d}{d}$ monomials.

Algebraic Circuit Complexity

Q: What is the most succinct way of representing the given polynomial of interest?

An n-variate, degree d-polynomial has $\binom{n+d}{d}$ monomials.

Thus representing a polynomial as a vector of coefficients requires $\binom{n+d}{d}$ size.

Algebraic Circuit Complexity

Q: What is the most succinct way of representing the given polynomial of interest?

An n-variate, degree d-polynomial has $\binom{n+d}{d}$ monomials.

Thus representing a polynomial as a vector of coefficients requires $\binom{n+d}{d}$ size.

Is there a representation that takes $\text{poly}(n, d)$ size?

Q: What is the most succinct way of representing the given polynomial of interest?

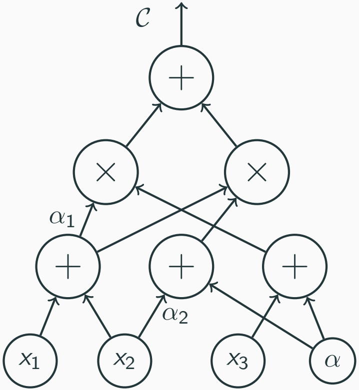
An n-variate, degree d-polynomial has $\binom{n+d}{d}$ monomials.

Thus representing a polynomial as a vector of coefficients requires $\binom{n+d}{d}$ size.

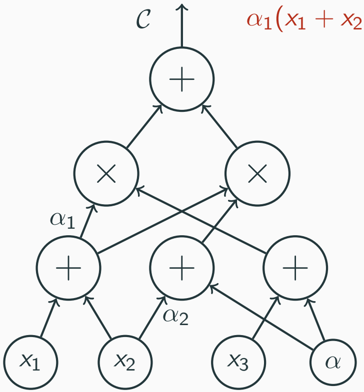
Is there a representation that takes $\text{poly}(n, d)$ size?

[Shamir 79, Lipton 94]: If $h(x) = \prod_{i=1}^{d}(x - i)$ can be computed using $\text{poly}(\log d)$ additions and multiplications, then integer factoring is easy for boolean circuits.

Algebraic Models of Computation



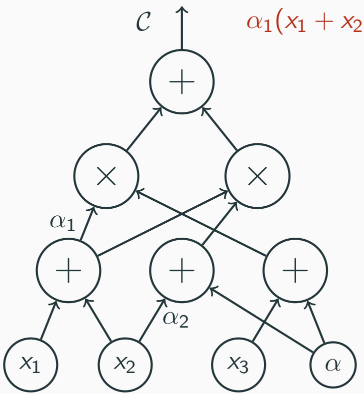
Algebraic Models of Computation

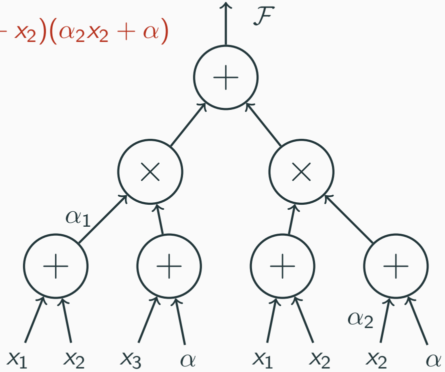


$\alpha_1(x_1 + x_2)(x_3 + \alpha) + (x_1 + x_2)(\alpha_2 x_2 + \alpha)$

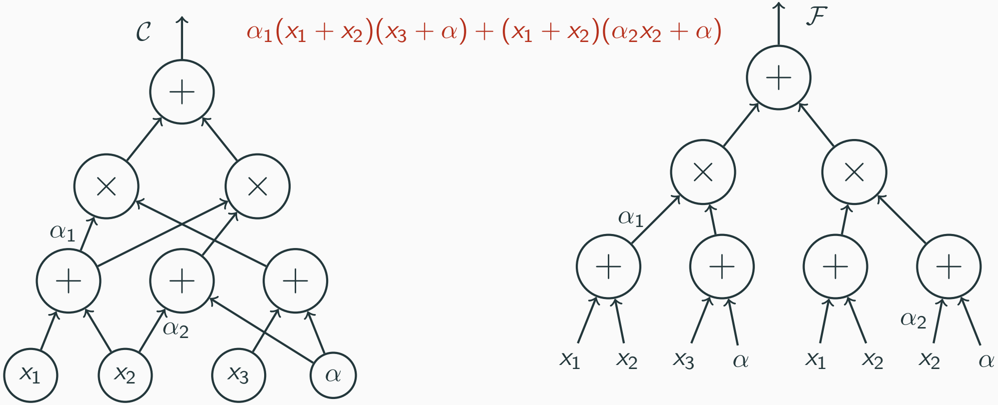
Algebraic Models of Computation

$$\alpha_1(x_1+x_2)(x_3+\alpha)+(x_1+x_2)(\alpha_2 x_2+\alpha)$$



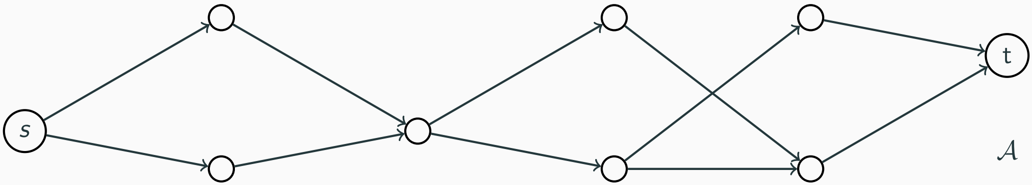


Algebraic Models of Computation

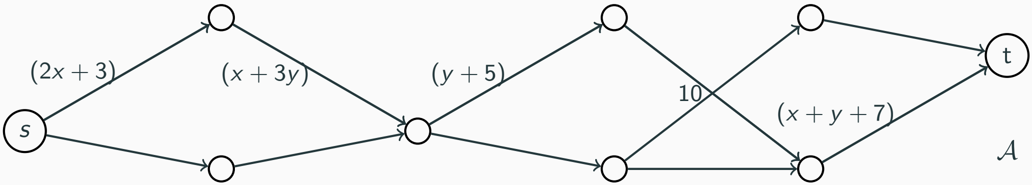


Q: Given $f(\mathbf{x}) \in \mathbb{F}[x_1, \ldots, x_n]$ of degree d, how many $+, \times, -$ gates are needed to compute f?

Algebraic Branching Programs

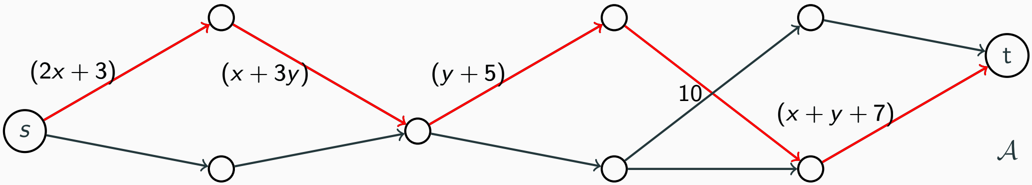


Algebraic Branching Programs



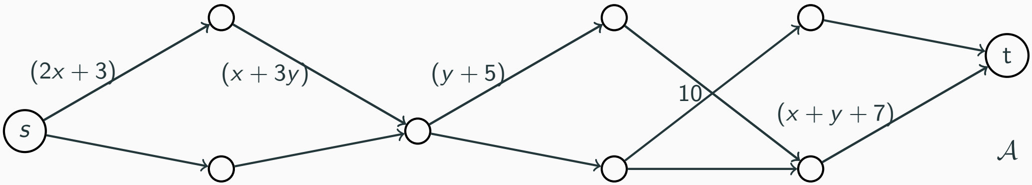
- Label on each edge: An affine linear form in $\{x_1, x_2, \ldots, x_n\}$

Algebraic Branching Programs



- Label on each edge: An affine linear form in $\{x_1, x_2, \ldots, x_n\}$
- Polynomial computed by the path $p = \mathrm{wt}(p)$: Product of the edge labels on p

Algebraic Branching Programs



- Label on each edge: An affine linear form in $\{x_1, x_2, \ldots, x_n\}$
- Polynomial computed by the path $p = \mathrm{wt}(p)$: Product of the edge labels on p
- Polynomial computed by the ABP: $f_{\mathcal{A}}(\mathbf{x}) = \sum_p \mathrm{wt}(p)$

Lower Bounds in Algebraic Circuit Complexity

Objects of Study: Polynomials over n variables of degree d.

Lower Bounds in Algebraic Circuit Complexity

Objects of Study: Polynomials over n variables of degree d.

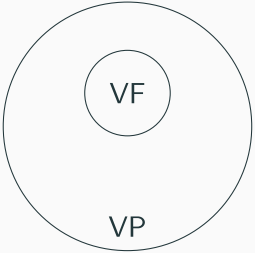
VP: Polynomials computable by circuits of size $\mathrm{poly}(n, d)$.

Lower Bounds in Algebraic Circuit Complexity

Objects of Study: Polynomials over n variables of degree d.

VF: Polynomials computable by formulas of size $\text{poly}(n, d)$.

VP: Polynomials computable by circuits of size $\text{poly}(n, d)$.

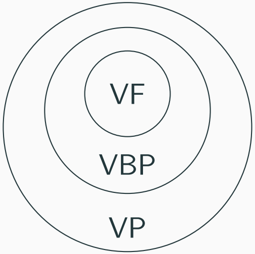


Objects of Study: Polynomials over n variables of degree d.

VF: Polynomials computable by formulas of size $\text{poly}(n, d)$.

VBP: Polynomials computable by ABPs of size $\text{poly}(n, d)$.

VP: Polynomials computable by circuits of size $\text{poly}(n, d)$.



Lower Bounds in Algebraic Circuit Complexity

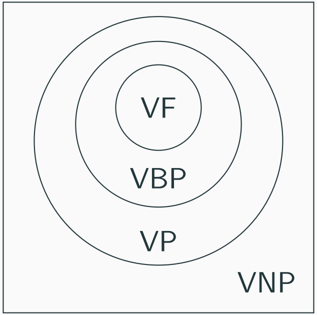
Objects of Study: Polynomials over n variables of degree d.

VF: Polynomials computable by formulas of size poly(n, d).

VBP: Polynomials computable by ABPs of size poly(n, d).

VP: Polynomials computable by circuits of size poly(n, d).

VNP: Explicit Polynomials $\left[\sum_{\mathbf{y} \in (0,1)^{\text{poly}(|\mathbf{x}|)}} \text{VP}(\mathbf{x}, \mathbf{y})\right]$



Lower Bounds in Algebraic Circuit Complexity

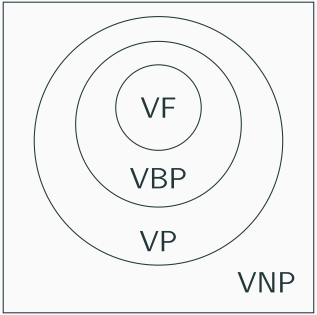
Objects of Study: Polynomials over n variables of degree d.

VF: Polynomials computable by formulas of size $\text{poly}(n, d)$.

VBP: Polynomials computable by ABPs of size $\text{poly}(n, d)$.

VP: Polynomials computable by circuits of size $\text{poly}(n, d)$.

VNP: Explicit Polynomials $\left[\sum_{\mathbf{y}\in(0,1)^{\text{poly}(|\mathbf{x}|)}} \text{VP}(\mathbf{x}, \mathbf{y})\right]$



Central Question: Find explicit polynomials that cannot be computed by efficient circuits.

Lower Bounds in Algebraic Circuit Complexity

Objects of Study: Polynomials over n variables of degree d.

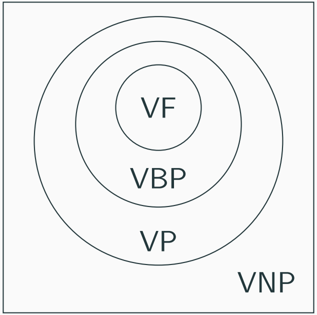
VF: Polynomials computable by formulas of size $\mathrm{poly}(n, d)$.

VBP: Polynomials computable by ABPs of size $\mathrm{poly}(n, d)$.

VP: Polynomials computable by circuits of size $\mathrm{poly}(n, d)$.

VNP: Explicit Polynomials $\left[\sum_{\mathbf{y}\in(0,1)^{\mathrm{poly}(|\mathbf{x}|)}} \mathrm{VP}(\mathbf{x}, \mathbf{y})\right]$

$$\boxed{\mathrm{VP} = \mathrm{VNP} \overset{\text{G.R.H.}}{\Longrightarrow} \mathrm{P} = \mathrm{NP}}$$



Central Question: Find explicit polynomials that cannot be computed by efficient circuits.

Lower Bounds in Algebraic Circuit Complexity

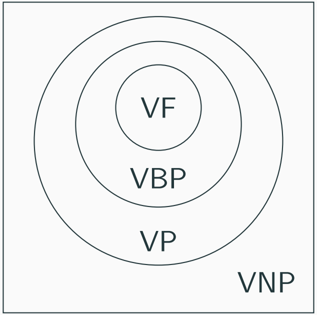
Objects of Study: Polynomials over n variables of degree d.

VF: Polynomials computable by formulas of size $\mathrm{poly}(n, d)$.

VBP: Polynomials computable by ABPs of size $\mathrm{poly}(n, d)$.

VP: Polynomials computable by circuits of size $\mathrm{poly}(n, d)$.

VNP: Explicit Polynomials $\left[\sum_{\mathbf{y} \in (0,1)^{\mathrm{poly}(|\mathbf{x}|)}} \mathrm{VP}(\mathbf{x}, \mathbf{y})\right]$



$$\boxed{\mathrm{VP} = \mathrm{VNP} \overset{\text{G.R.H.}}{\Longrightarrow} \mathrm{P} = \mathrm{NP}}$$

Central Question: Find explicit polynomials that cannot be computed by efficient circuits.

Other Motivating Questions: Are the other inclusions tight?

Lower Bounds for General Models

General Circuits

[Baur-Strassen 83]: Any algebraic circuit computing $\sum_{i=1}^{n} x_i^d$ requires $\Omega(n \log d)$ wires.

Lower Bounds for General Models

General Circuits

[Baur-Strassen 83]: Any algebraic circuit computing $\sum_{i=1}^{n} x_i^d$ requires $\Omega(n \log d)$ wires.

General ABPs

[C-Kumar-She-Volk 22]: Any ABP computing $\sum_{i=1}^{n} x_i^d$ requires $\Omega(nd)$ vertices.

Lower Bounds for General Models

General Circuits

[Baur-Strassen 83]: Any algebraic circuit computing $\sum_{i=1}^{n} x_i^d$ requires $\Omega(n \log d)$ wires.

General ABPs

[C-Kumar-She-Volk 22]: Any ABP computing $\sum_{i=1}^{n} x_i^d$ requires $\Omega(nd)$ vertices.

General Formulas

[Kalorkoti 85]: Any formula computing the n^2-variate $\mathrm{Det}_n(\mathbf{x})$ requires $\Omega(n^3)$ wires.

Lower Bounds for General Models

General Circuits

[Baur-Strassen 83]: Any algebraic circuit computing $\sum_{i=1}^{n} x_i^d$ requires $\Omega(n \log d)$ wires.

General ABPs

[C-Kumar-She-Volk 22]: Any ABP computing $\sum_{i=1}^{n} x_i^d$ requires $\Omega(nd)$ vertices.

General Formulas

[Kalorkoti 85]: Any formula computing the n^2-variate $\mathrm{Det}_n(\mathbf{x})$ requires $\Omega(n^3)$ wires.

[Shpilka-Yehudayoff 10] (using Kalorkoti's method): There is an n-variate multilinear polynomial such that any formula computing it requires $\Omega(n^2/\log n)$ wires.

Lower Bounds for General Models

General Circuits

[Baur-Strassen 83]: Any algebraic circuit computing $\sum_{i=1}^{n} x_i^d$ requires $\Omega(n \log d)$ wires.

General ABPs

[C-Kumar-She-Volk 22]: Any ABP computing $\sum_{i=1}^{n} x_i^d$ requires $\Omega(nd)$ vertices.

General Formulas

[Kalorkoti 85]: Any formula computing the n^2-variate $\mathrm{Det}_n(\mathbf{x})$ requires $\Omega(n^3)$ wires.

[Shpilka-Yehudayoff 10] (using Kalorkoti's method): There is an n-variate multilinear polynomial such that any formula computing it requires $\Omega(n^2/\log n)$ wires.

[C-Kumar-She-Volk 22]: Any formula computing $\mathrm{ESYM}_{n,0.1n}(\mathbf{x})$ requires $\Omega(n^2)$ vertices.

$$\mathrm{ESYM}_{n,d}(\mathbf{x}) = \sum_{i_1 < \cdots < i_d \in [n]} x_{i_1} \cdots x_{i_d}.$$

Towards Better ABP Lower Bounds

[C-Kumar-She-Volk 22]: Any ABP computing $\sum_{i=1}^{n} x_i^d$ requires $\Omega(nd)$ vertices.

Towards Better ABP Lower Bounds

[C-Kumar-She-Volk 22]: Any ABP computing $\sum_{i=1}^{n} x_i^d$ requires $\Omega(nd)$ vertices.

[Bhargav-Dwivedi-Saxena 24]: Super polynomial lower bound against total-width of $\sum$ osmABP for a polynomial of degree $d = O\left(\frac{\log n}{\log \log n}\right) \implies$ super-polynomial lower bound against ABPs.

Towards Better ABP Lower Bounds

[C-Kumar-She-Volk 22]: Any ABP computing $\sum_{i=1}^{n} x_i^d$ requires $\Omega(nd)$ vertices.

[Bhargav-Dwivedi-Saxena 24]: Super polynomial lower bound against total-width of $\sum$osmABP for a polynomial of degree $d = O\left(\frac{\log n}{\log \log n}\right) \implies$ super-polynomial lower bound against ABPs.

[C-Kush-Saraf-Shpilka 24]: For $\omega(\log n) = d \leq n$, there is a polynomial $G_{n,d}(\mathbf{x})$ which is set-multilinear w.r.t $\mathbf{x} = \{\mathbf{x}_1, \ldots, \mathbf{x}_d\}$, where $|\mathbf{x}_i| \leq n$ for every $i \in [d]$, such that:

- $G_{n,d}$ is computable by a set-multilinear ABP of size $\text{poly}(n)$,
- any $\sum$osmABP computing $G_{n,d}$ must have super-polynomial total-width.

Non-Commutativity

$$f(x, y) = (x + y) \times (x + y) = x^2 + xy + yx + y^2 \neq x^2 + 2xy + y^2$$

Non-Commutativity

$$f(x,y) = (x+y) \times (x+y) = x^2 + xy + yx + y^2 \neq x^2 + 2xy + y^2$$

Non-Commutative Models: The multiplication gates are assumed to be non-commutative.

Non-Commutativity

$$f(x, y) = (x + y) \times (x + y) = x^2 + xy + yx + y^2 \neq x^2 + 2xy + y^2$$

Non-Commutative Models: The multiplication gates are assumed to be non-commutative.

Can we do better in this setting?

Non-Commutativity

$$f(x, y) = (x + y) \times (x + y) = x^2 + xy + yx + y^2 \neq x^2 + 2xy + y^2$$

Non-Commutative Models: The multiplication gates are assumed to be non-commutative.

Can we do better in this setting? For general circuits, continues to be $\Omega(n \log d)$.

Non-Commutativity

$$f(x,y) = (x+y)\times(x+y) = x^2 + xy + yx + y^2 \neq x^2 + 2xy + y^2$$

Non-Commutative Models: The multiplication gates are assumed to be non-commutative.

Can we do better in this setting? For general circuits, continues to be $\Omega(n \log d)$.

[C-Hrubeš 23]: Any homogeneous non-commutative circuit computing

$$\mathrm{OSym}_{n,d}(\mathbf{x}) = \sum_{1\leq i_1<\cdots<i_d\leq n} x_{i_1}\cdots x_{i_d}$$

has size $\Omega(nd)$ for $d \leq \frac{n}{2}$.

Non-Commutativity

$$f(x,y) = (x+y)\times(x+y) = x^2 + xy + yx + y^2 \neq x^2 + 2xy + y^2$$

Non-Commutative Models: The multiplication gates are assumed to be non-commutative.

Can we do better in this setting? For general circuits, continues to be $\Omega(n \log d)$.

[C-Hrubeš 23]: Any homogeneous non-commutative circuit computing

$$\mathrm{OSym}_{n,d}(\mathbf{x}) = \sum_{1\le i_1<\cdots<i_d\le n} x_{i_1}\cdots x_{i_d}$$

has size $\Omega(nd)$ for $d \le \frac{n}{2}$. The lower bound is tight for homogeneous non-commutative circuits.

Non-Commutativity

$$f(x,y) = (x+y)\times(x+y) = x^2 + xy + yx + y^2 \neq x^2 + 2xy + y^2$$

Non-Commutative Models: The multiplication gates are assumed to be non-commutative.

Can we do better in this setting? For general circuits, continues to be $\Omega(n \log d)$.

[C-Hrubeš 23]: Any homogeneous non-commutative circuit computing

$$\mathrm{OSym}_{n,d}(\mathbf{x}) = \sum_{1\le i_1<\cdots<i_d\le n} x_{i_1}\cdots x_{i_d}$$

has size $\Omega(nd)$ for $d \le \frac{n}{2}$. The lower bound is tight for homogeneous non-commutative circuits.

Further, there is a non-commutative circuit of size $O(n \log^2 n)$ that computes $\mathrm{OSym}_{n,n/2}(\mathbf{x})$.

ABP vs Formula in the Non-Commutative Setting

[Nisan 91]: Any ABP computing $\mathrm{Pal}_n(x_0, x_1) = \sum_{w \in \{0,1\}^{n/2}} \mathbf{x}_w \cdot \mathbf{x}_{w^R}$ has size $2^{\Omega(n)}$.

ABP vs Formula in the Non-Commutative Setting

[Nisan 91]: Any ABP computing $\mathrm{Pal}_n(x_0, x_1) = \sum_{w \in \{0,1\}^{n/2}} \mathbf{x}_w \cdot \mathbf{x}_{w^R}$ has size $2^{\Omega(n)}$.

[Tavenas-Limaye-Srinivasan 22]: Any homogeneous non-commutative formula computing $\mathrm{IMM}_{n,d}(\mathbf{x})$ has size $n^{\Omega(\log \log d)}$.

ABP vs Formula in the Non-Commutative Setting

[Nisan 91]: Any ABP computing $\mathrm{Pal}_n(x_0, x_1) = \sum_{w \in \{0,1\}^{n/2}} \mathbf{x}_w \cdot \mathbf{x}_{w^R}$ has size $2^{\Omega(n)}$.

[Tavenas-Limaye-Srinivasan 22]: Any homogeneous non-commutative formula computing $\mathrm{IMM}_{n,d}(\mathbf{x})$ has size $n^{\Omega(\log \log d)}$.

[**Cha** 21]: For $\mathbf{x} = \cup_{i \in [n]} \{X_i\}$ with $X_i = \{x_{i,j}\}_{j \in [n]}$,

ABP vs Formula in the Non-Commutative Setting

[Nisan 91]: Any ABP computing $\mathrm{Pal}_n(x_0, x_1) = \sum_{w \in \{0,1\}^{n/2}} \mathbf{x}_w \cdot \mathbf{x}_{w^R}$ has size $2^{\Omega(n)}$.

[Tavenas-Limaye-Srinivasan 22]: Any homogeneous non-commutative formula computing $\mathrm{IMM}_{n,d}(\mathbf{x})$ has size $n^{\Omega(\log \log d)}$.

[**Cha** **21**]: For $\mathbf{x} = \cup_{i \in [n]} \{X_i\}$ with $X_i = \{x_{i,j}\}_{j \in [n]}$, there exists a $(\log n)$-degree abecedarian polynomial $f \in \mathbb{F}\langle \mathbf{x} \rangle$ that is computable by an ABP of size $O(nd)$ such that

ABP vs Formula in the Non-Commutative Setting

[Nisan 91]: Any ABP computing $\mathrm{Pal}_n(x_0, x_1) = \sum_{w \in \{0,1\}^{n/2}} \mathbf{x}_w \cdot \mathbf{x}_{w^R}$ has size $2^{\Omega(n)}$.

[Tavenas-Limaye-Srinivasan 22]: Any homogeneous non-commutative formula computing $\mathrm{IMM}_{n,d}(\mathbf{x})$ has size $n^{\Omega(\log \log d)}$.

[**Cha** **21**]: For $\mathbf{x} = \cup_{i \in [n]} \{X_i\}$ with $X_i = \{x_{i,j}\}_{j \in [n]}$, there exists a $(\log n)$-degree abecedarian polynomial $f \in \mathbb{F}\langle \mathbf{x} \rangle$ that is computable by an ABP of size $O(nd)$ such that

- Any abecedarian formula computing f has size $n^{\Omega(\log \log n)}$.

ABP vs Formula in the Non-Commutative Setting

[Nisan 91]: Any ABP computing $\mathrm{Pal}_n(x_0, x_1) = \sum_{w \in \{0,1\}^{n/2}} \mathbf{x}_w \cdot \mathbf{x}_{w^R}$ has size $2^{\Omega(n)}$.

[Tavenas-Limaye-Srinivasan 22]: Any homogeneous non-commutative formula computing $\mathrm{IMM}_{n,d}(\mathbf{x})$ has size $n^{\Omega(\log \log d)}$.

[**Cha** **21**]: For $\mathbf{x} = \cup_{i\in[n]} \{X_i\}$ with $X_i = \{x_{i,j}\}_{j\in[n]}$, there exists a $(\log n)$-degree abecedarian polynomial $f \in \mathbb{F}\langle \mathbf{x} \rangle$ that is computable by an ABP of size $O(nd)$ such that

- Any abecedarian formula computing f has size $n^{\Omega(\log \log n)}$.
- There is an abecedarian formula of size $n^{O(\log \log n)}$ that computes f.

ABP vs Formula in the Non-Commutative Setting

[Nisan 91]: Any ABP computing $\mathrm{Pal}_n(x_0, x_1) = \sum_{w \in \{0,1\}^{n/2}} \mathbf{x}_w \cdot \mathbf{x}_{w^R}$ has size $2^{\Omega(n)}$.

[Tavenas-Limaye-Srinivasan 22]: Any homogeneous non-commutative formula computing $\mathrm{IMM}_{n,d}(\mathbf{x})$ has size $n^{\Omega(\log \log d)}$.

[**Cha** **21**]: For $\mathbf{x} = \cup_{i \in [n]} \{X_i\}$ with $X_i = \{x_{i,j}\}_{j \in [n]}$, there exists a $(\log n)$-degree abecedarian polynomial $f \in \mathbb{F}\langle \mathbf{x} \rangle$ that is computable by an ABP of size $O(nd)$ such that

- Any abecedarian formula computing f has size $n^{\Omega(\log \log n)}$.
- There is an abecedarian formula of size $n^{O(\log \log n)}$ that computes f.

If an n-variate polynomial is abecedarian with respect to $\{X_1, \ldots, X_m\}$ for $m = \log n$,

ABP vs Formula in the Non-Commutative Setting

[Nisan 91]: Any ABP computing $\mathrm{Pal}_n(x_0, x_1) = \sum_{w \in \{0,1\}^{n/2}} \mathbf{x}_w \cdot \mathbf{x}_{w^R}$ has size $2^{\Omega(n)}$.

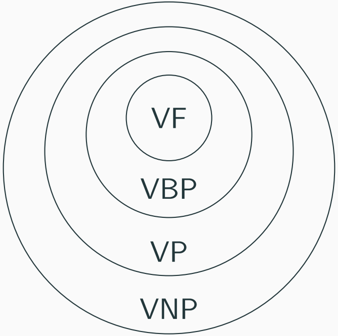
[Tavenas-Limaye-Srinivasan 22]: Any homogeneous non-commutative formula computing $\mathrm{IMM}_{n,d}(\mathbf{x})$ has size $n^{\Omega(\log \log d)}$.

[**Cha** **21**]: For $\mathbf{x} = \cup_{i \in [n]} \{X_i\}$ with $X_i = \{x_{i,j}\}_{j \in [n]}$, there exists a $(\log n)$-degree abecedarian polynomial $f \in \mathbb{F}\langle \mathbf{x} \rangle$ that is computable by an ABP of size $O(nd)$ such that

- Any abecedarian formula computing f has size $n^{\Omega(\log \log n)}$.
- There is an abecedarian formula of size $n^{O(\log \log n)}$ that computes f.

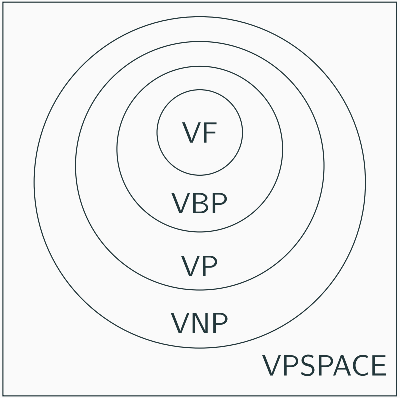
If an n-variate polynomial is abecedarian with respect to $\{X_1, \ldots, X_m\}$ for $m = \log n$, then any formula computing f can be made abecedarian with only $\mathrm{poly}(n)$ blow-up in size.

Classes Beyond VNP



Classes Beyond VNP

$\mathsf{VPSPACE}_b$: Polynomials whose coefficients can be computed in PSPACE/poly and have degree bounded by poly(n).

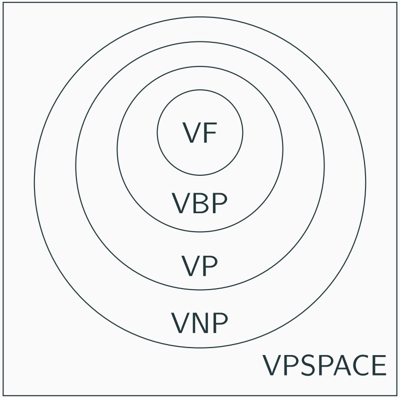


Classes Beyond VNP

$\mathrm{VPSPACE}_b$: Polynomials whose coefficients can be computed in PSPACE/poly and have degree bounded by $\mathrm{poly}(n)$.

[Koiran-Perifel 09]

$$\mathrm{VNP} \neq \mathrm{VPSPACE}_b \implies \mathrm{P/poly} \neq \mathrm{PSPACE/poly}.$$



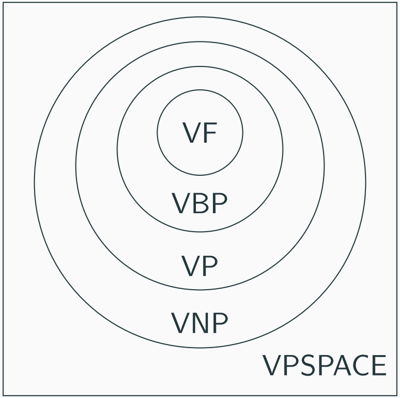
Classes Beyond VNP

$\mathrm{VPSPACE}_b$: Polynomials whose coefficients can be computed in PSPACE/poly and have degree bounded by poly(n).

[Koiran-Perifel 09]

$$\mathrm{VNP} \neq \mathrm{VPSPACE}_b \implies \mathrm{P/poly} \neq \mathrm{PSPACE/poly}.$$

$$\boxed{\mathrm{VNP} \stackrel{?}{=} \mathrm{VPSPACE}_b}$$



Classes Beyond VNP

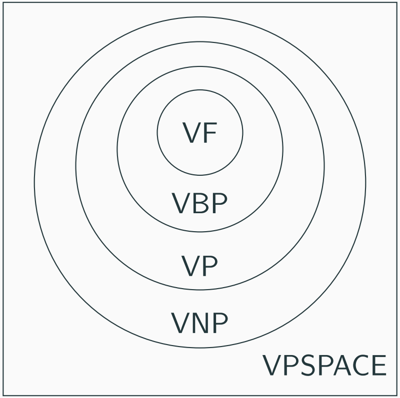
$\mathsf{VPSPACE}_b$: Polynomials whose coefficients can be computed in PSPACE/poly and have degree bounded by poly(n).

[Koiran-Perifel 09]

$$\mathsf{VNP} \neq \mathsf{VPSPACE}_b \implies \mathsf{P/poly} \neq \mathsf{PSPACE/poly}.$$

$$\boxed{\mathsf{VNP} \stackrel{?}{=} \mathsf{VPSPACE}_b}$$

[C-Gajjar-Tengse 24]: $\mathsf{VNP} \neq \mathsf{VPSPACE}_b$ in the monotone setting.



Proving Lower Bounds: Finding a Measure

1. Build a function $\Gamma : \mathbb{F}[\mathbf{x}] \to \mathbb{N}$.

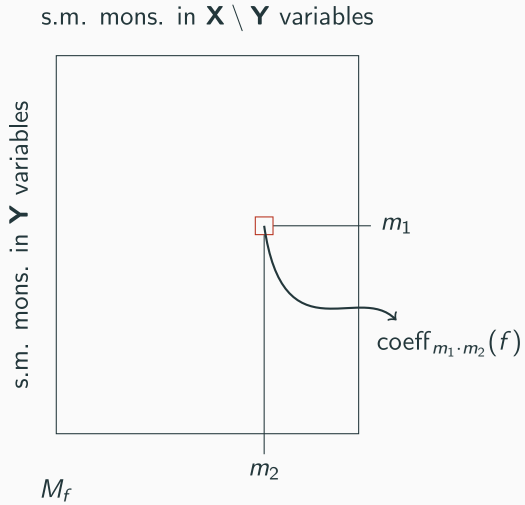
Proving Lower Bounds: Finding a Measure

1. Build a function $\Gamma : \mathbb{F}[\mathbf{x}] \to \mathbb{N}$.
2. Show that if a polynomial is computable efficiently by the model of choice, then $\Gamma(f)$ must be small.

Proving Lower Bounds: Finding a Measure

1. Build a function $\Gamma : \mathbb{F}[\mathbf{x}] \to \mathbb{N}$.
2. Show that if a polynomial is computable efficiently by the model of choice, then $\Gamma(f)$ must be small.
3. Find an explicit polynomial f such that $\Gamma(f)$ is large.

Proving Lower Bounds: Finding a Measure

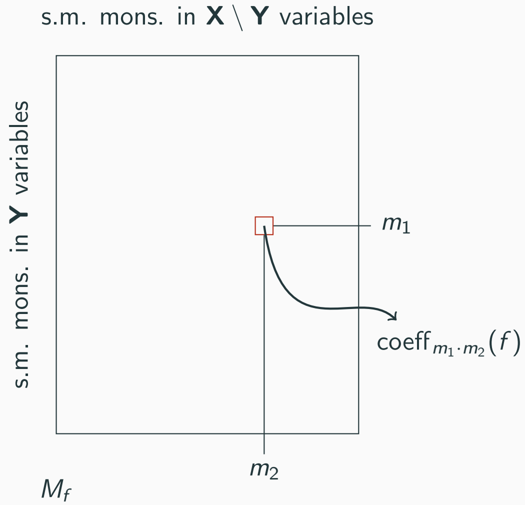


1. Build a function $\Gamma : \mathbb{F}[\mathbf{x}] \to \mathbb{N}$.
2. Show that if a polynomial is computable efficiently by the model of choice, then $\Gamma(f)$ must be small.
3. Find an explicit polynomial f such that $\Gamma(f)$ is large.

Note: Γ is almost always the dimension of some algebraic object and most of the time is simply the rank of a matrix associated with f.

.

Proving Lower Bounds: Finding a Measure



1. Build a function $\Gamma : \mathbb{F}[\mathbf{x}] \to \mathbb{N}$.
2. Show that if a polynomial is computable efficiently by the model of choice, then $\Gamma(f)$ must be small.
3. Find an explicit polynomial f such that $\Gamma(f)$ is large.

Note: Γ is almost always the dimension of some algebraic object and most of the time is simply the rank of a matrix associated with f.

The property "a matrix has low-rank" can be captured by a polynomial equation.

Natural Proofs

Natural Proofs in the Algebraic Setting

$0 \neq Q \in \mathbb{F}[\mathbf{y}]$ such that $Q(\text{coeff-vector of } f) = 0$ for every f that is computable efficiently by the model of interest.

Natural Proofs

Natural Proofs in the Algebraic Setting

$0 \neq Q \in \mathbb{F}[\mathbf{y}]$ such that $Q(\text{coeff-vector of } f) = 0$ for every f that is computable efficiently by the model of interest.

Question: What is the complexity of Q when the model of interest is VP?

Natural Proofs

Natural Proofs in the Algebraic Setting

$0 \neq Q \in \mathbb{F}[\mathbf{y}]$ such that $Q(\text{coeff-vector of } f) = 0$ for every f that is computable efficiently by the model of interest.

Question: What is the complexity of Q when the model of interest is VP? Is $Q \in \text{VP}$?

Natural Proofs

Natural Proofs in the Algebraic Setting

$0 \neq Q \in \mathbb{F}[\mathbf{y}]$ such that $Q(\text{coeff-vector of } f) = 0$ for every f that is computable efficiently by the model of interest.

Question: What is the complexity of Q when the model of interest is VP? Is $Q \in \text{VP}$?

[Forbes-Shpilka-Volk 18, Grochow-Kumar-Saks-Saraf]: Under some assumptions, no!

Natural Proofs

Natural Proofs in the Algebraic Setting

$0 \neq Q \in \mathbb{F}[\mathbf{y}]$ such that $Q(\text{coeff-vector of } f) = 0$ for every f that is computable efficiently by the model of interest.

Question: What is the complexity of Q when the model of interest is VP? Is $Q \in \mathrm{VP}$?

[Forbes-Shpilka-Volk 18, Grochow-Kumar-Saks-Saraf]: Under some assumptions, no!

[C-Kumar-Ramya-Saptharishi-Tengse 20]: Yes if the model of interest is $\mathrm{VP}_{\text{bd-coeff}}$.

Natural Proofs in the Algebraic Setting

$0 \neq Q \in \mathbb{F}[\mathbf{y}]$ such that $Q(\text{coeff-vector of } f) = 0$ for every f that is computable efficiently by the model of interest.

Question: What is the complexity of Q when the model of interest is VP? Is $Q \in$ VP?

[Forbes-Shpilka-Volk 18, Grochow-Kumar-Saks-Saraf]: Under some assumptions, no!

[C-Kumar-Ramya-Saptharishi-Tengse 20]: Yes if the model of interest is $\mathsf{VP}_{\text{bd-coeff}}$.

[K-R-S-T 22]: Perm_n is optimally hard $\implies$ no if the model of interest is VNP.

Natural Proofs in the Algebraic Setting

$0 \neq Q \in \mathbb{F}[\mathbf{y}]$ such that $Q(\text{coeff-vector of } f) = 0$ for every f that is computable efficiently by the model of interest.

Question: What is the complexity of Q when the model of interest is VP? Is $Q \in \text{VP}$?

[Forbes-Shpilka-Volk 18, Grochow-Kumar-Saks-Saraf]: Under some assumptions, no!

[C-Kumar-Ramya-Saptharishi-Tengse 20]: Yes if the model of interest is $\text{VP}_{\text{bd-coeff}}$.

[K-R-S-T 22]: Perm_n is optimally hard $\implies$ no if the model of interest is VNP.

[C-Tengse]: $Q \in \text{VPSPACE}[\log^{\log^*}]$.

Ongoing and Future Projects

Some Open Directions

- Better lower bounds against homogeneous formulas?

Some Open Directions

- Better lower bounds against homogeneous formulas?
- Super-linear Lower Bounds against ABPs for constant degree polynomials?

Some Open Directions

- Better lower bounds against homogeneous formulas?
- Super-linear Lower Bounds against ABPs for constant degree polynomials?
- Super-linear Lower Bounds against Determinantal Complexity?

Some Open Directions

- Better lower bounds against homogeneous formulas?
- Super-linear Lower Bounds against ABPs for constant degree polynomials?
- Super-linear Lower Bounds against Determinantal Complexity?
- Better lower bounds against set-multilinear ABPs?

Some Open Directions

- Better lower bounds against homogeneous formulas?
- Super-linear Lower Bounds against ABPs for constant degree polynomials?
- Super-linear Lower Bounds against Determinantal Complexity?
- Better lower bounds against set-multilinear ABPs?
- Better Lower Bounds against Non-Commutative circuits?

Some Open Directions

- Better lower bounds against homogeneous formulas?
- Super-linear Lower Bounds against ABPs for constant degree polynomials?
- Super-linear Lower Bounds against Determinantal Complexity?
- Better lower bounds against set-multilinear ABPs?
- Better Lower Bounds against Non-Commutative circuits?
- Separating formulas and ABPs in the non-commutative setting?

Some Open Directions

- Better lower bounds against homogeneous formulas?
- Super-linear Lower Bounds against ABPs for constant degree polynomials?
- Super-linear Lower Bounds against Determinantal Complexity?
- Better lower bounds against set-multilinear ABPs?
- Better Lower Bounds against Non-Commutative circuits?
- Separating formulas and ABPs in the non-commutative setting?
- Do VP have VP natural proofs under some reasonable conditions?

Branching Out

Study complexity theoretic questions about Boolean Circuits, Communication Models.

Teaching etc.

Courses I would be happy to teach

Existing Courses

- MA5310: Linear Algebra
- MA5320: Algebra I
- MA5330: Real Analysis
- MA5400: Probability Theory
- MA5380: Topology
- MA5741: Object Oriented Programming
- MA5910: Data Structures and Algorithms
- MA6200: Theory of Computation

Courses I would be happy to teach

Existing Courses

- MA5310: Linear Algebra
- MA5320: Algebra I
- MA5330: Real Analysis
- MA5400: Probability Theory
- MA5380: Topology
- MA5741: Object Oriented Programming
- MA5910: Data Structures and Algorithms
- MA6200: Theory of Computation

Additional Courses

- Randomness in Computation
- Pseudorandomness
- Algebra and Computation
- Computational Complexity Theory
- Communication Complexity
- Secure Computation
- Circuit Complexity
- Algebraic Complexity Theory

Courses I would be happy to teach

Existing Courses

- MA5310: Linear Algebra
- MA5320: Algebra I
- MA5330: Real Analysis
- MA5400: Probability Theory
- MA5380: Topology
- MA5741: Object Oriented Programming
- MA5910: Data Structures and Algorithms
- MA6200: Theory of Computation

Additional Courses

- Randomness in Computation
- Pseudorandomness
- Algebra and Computation
- Computational Complexity Theory
- Communication Complexity
- Secure Computation
- Circuit Complexity
- Algebraic Complexity Theory

I would be happy to teach/design other courses depending on interest and/or requirement.

Outreach Activities

Greatly passionate about outreach: I would love to help organise outreach programmes.

Outreach Activities

Greatly passionate about outreach: I would love to help organise outreach programmes.

- Invited to give talks at Curry Leaf Days (MTTS Alumni Initiative) and the Summer School for Women in Mathematics and Statistics (2024).
- Helped in organizing STCS Vigyan Vidushi at TIFR Mumbai during my final year of PhD (2021). Was part of the mentoring session in the 2022, 2024 editions.
- Invited to give a talk at the CSA Summer School, IISc Bangalore (2019).
- Tutor at the Summer School for Women in Mathematics and Statistics (2019).
- Member of the outreach team of the STCS and TIFR during PhD (2018-22).

Outreach Activities

Greatly passionate about outreach: I would love to help organise outreach programmes.

- Invited to give talks at Curry Leaf Days (MTTS Alumni Initiative) and the Summer School for Women in Mathematics and Statistics (2024).
- Helped in organizing STCS Vigyan Vidushi at TIFR Mumbai during my final year of PhD (2021). Was part of the mentoring session in the 2022, 2024 editions.
- Invited to give a talk at the CSA Summer School, IISc Bangalore (2019).
- Tutor at the Summer School for Women in Mathematics and Statistics (2019).
- Member of the outreach team of the STCS and TIFR during PhD (2018-22).

I would also like to help organise seminars, webinars and workshops to introduce the students to the recent trends of research.

Outreach Activities

Greatly passionate about outreach: I would love to help organise outreach programmes.

- Invited to give talks at Curry Leaf Days (MTTS Alumni Initiative) and the Summer School for Women in Mathematics and Statistics (2024).
- Helped in organizing STCS Vigyan Vidushi at TIFR Mumbai during my final year of PhD (2021). Was part of the mentoring session in the 2022, 2024 editions.
- Invited to give a talk at the CSA Summer School, IISc Bangalore (2019).
- Tutor at the Summer School for Women in Mathematics and Statistics (2019).
- Member of the outreach team of the STCS and TIFR during PhD (2018-22).

I would also like to help organise seminars, webinars and workshops to introduce the students to the recent trends of research. Restart student seminar?

Thank you!!!