Arithmetic Complexity and Polynomial Identity Testing

Manindra Agrawal

IIT Kanpur

Mysore Park Workshop, 2012

Workshop on Complexity and Logic, August 17-19, IIT Kanpur

- To celebrate 60th birthday of Somenath Biswas.
- Speakers: Eric Allender, Jaikumar Radhakrishnan, Nutan Limaye, Jaylal Sharma, Prahlad Harsha, Osamu Watanabe, Bruno Poizat, Markus Blaeser, Ragesh Jaiswal, Meena Mahajan, K V Subramanian
- Website: http://www.cse.iitk.ac.in/users/ppk/wcl/
- Interested participants should send a mail to `manindra@iitk.ac.in` or `ppk@iitk.ac.in`.

Overview

Outline

Polynomials

Definition

$P(x_1, x_2, \ldots, x_n)$ is a polynomial of variables x_1, x_2, ..., x_n and degree d over field F if:

$$P(x_1, x_2, \ldots, x_n) = \sum_{0 \leq i_1, i_2, \ldots, i_n \leq d} c_{i_1, i_2, \ldots, i_n} x_1^{i_1} x_2^{i_2} \cdots x_n^{i_n}$$

where $c_{i_1, i_2, \ldots, i_n} \in F$.

- Polynomials are one of the fundamental objects in mathematics.
- Special polynomials and their properties play a crucial role in a number of branches.

Polynomials

Definition

$P(x_1, x_2, \ldots, x_n)$ is a polynomial of variables x_1, x_2, …, x_n and degree d over field F if:

$$P(x_1, x_2, \ldots, x_n) = \sum_{0 \le i_1, i_2, \ldots, i_n \le d} c_{i_1, i_2, \ldots, i_n} x_1^{i_1} x_2^{i_2} \cdots x_n^{i_n}$$

where $c_{i_1, i_2, \ldots, i_n} \in F$.

- Polynomials are one of the fundamental objects in mathematics.
- Special polynomials and their properties play a crucial role in a number of branches.

Example: Chebyshev Polynomials

$$T_d(x) = \sum_{k=0}^{\lfloor d/2 \rfloor} \binom{d}{2k} (x^2 - 1)^k x^{d-2k}.$$

- Solutions of the differential equation

$$(1-x)^2 y'' - xy' + d^2 y = 0.$$

- Roots are $\cos(\frac{\pi}{2}\frac{(2k-1)}{d})$, $k = 1, \ldots, d$.
- Bounded within $[-1, 1]$ in the interval $[-1, 1]$, and used in interpolation.

Euler Polynomials

- Euler function is defined as:

$$E(x) = \prod_{k>0} (1 - x^k).$$

- This is a power series and can be alternately represented as the uniformly convergent limit of the polynomial family:

$$E_d(x) = \prod_{k=1}^{d} (1 - x^k)$$

in the open disk $|x| < 1$.

- Euler function is 'canonical' example of modular functions and is generating function of partition numbers:

$$\frac{1}{E(x)} = \prod_{k>0} \sum_{j \geq 0} x^{jk} = \sum_{j \geq 0} \Pi_j x^j$$

where Π_j equals the number of partitions of j.

Euler Polynomials

- Euler function is defined as:

$$E(x) = \prod_{k>0} (1 - x^k).$$

- This is a power series and can be alternately represented as the uniformly convergent limit of the polynomial family:

$$E_d(x) = \prod_{k=1}^{d} (1 - x^k)$$

  in the open disk $|x| < 1$.

- Euler function is 'canonical' example of modular functions and is generating function of partition numbers:

$$\frac{1}{E(x)} = \prod_{k>0} \sum_{j \geq 0} x^{jk} = \sum_{j \geq 0} \Pi_j x^j$$

  where Π_j equals the number of partitions of j.

Euler Polynomials

- Euler function is defined as:

$$E(x) = \prod_{k>0} (1 - x^k).$$

- This is a power series and can be alternately represented as the uniformly convergent limit of the polynomial family:

$$E_d(x) = \prod_{k=1}^{d} (1 - x^k)$$

in the open disk $|x| < 1$.

- Euler function is 'canonical' example of modular functions and is generating function of partition numbers:

$$\frac{1}{E(x)} = \prod_{k>0} \sum_{j\geq 0} x^{jk} = \sum_{j\geq 0} \Pi_j x^j$$

where Π_j equals the number of partitions of j.

Example: Determinant Polynomials

$$
\begin{aligned}
\mathsf{det}_n &= \mathsf{det} \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,n} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n,1} & x_{n,2} & \cdots & x_{n,n} \end{bmatrix} \\
&= \sum_{\pi \in S_n} \mathsf{sgn}(\pi) \cdot \prod_{i=1}^{n} x_{i,\pi(i)}.
\end{aligned}
$$

- Polynomial over n^2 variables, S_n is the set of all permutations over $\{1, 2, \ldots, n\}$, and $\mathsf{sgn}(\pi) \in \{-1, 1\}$.
- Roots are precisely all singular matrices.
- Linear function on rows and columns.

Example: Permanent Polynomials

$$\begin{aligned} \mathrm{per}_n &= \mathrm{per} \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,n} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n,1} & x_{n,2} & \cdots & x_{n,n} \end{bmatrix} \\ &= \sum_{\pi \in S_n} \prod_{i=1}^{n} x_{i,\pi(i)}. \end{aligned}$$

- Polynomial over n^2 variables, S_n is the set of all permutations over $\{1, 2, \ldots, n\}$.
- Similar to Determinant Polynomial, but has very different properties.
- Counts the number of perfect matchings in a bipartite graph.

Outline

Classifying Polynomials

- Polynomials are typically classified based on number of variables, degree, and number of non-zero terms.
- These classifications, however, do not capture differences in properties of polynomials well:
  - Polynomials

$$\prod_{k=1}^{n} x_k \quad \& \quad \prod_{k=1}^{n}(1+x_k)$$

    have similar properties but differ (hugely) in number of non-zero terms.
  - Polynomials $\det_n$ and per_n have same degree, number of variables, and number of non-zero terms, but have very different properties.

Classifying Polynomials

- Polynomials are typically classified based on number of variables, degree, and number of non-zero terms.
- These classifications, however, do not capture differences in properties of polynomials well:
  - Polynomials
    $$\prod_{k=1}^{n} x_k \ \& \ \prod_{k=1}^{n}(1+x_k)$$
    have similar properties but differ (hugely) in number of non-zero terms.
  - Polynomials det_n and per_n have same degree, number of variables, and number of non-zero terms, but have very different properties.

Classifying Polynomials

- Polynomials are typically classified based on number of variables, degree, and number of non-zero terms.
- These classifications, however, do not capture differences in properties of polynomials well:
    - Polynomials
    $$\prod_{k=1}^{n} x_k \;\;\&\;\; \prod_{k=1}^{n}(1+x_k)$$
    have similar properties but differ (hugely) in number of non-zero terms.
    - Polynomials det_n and per_n have same degree, number of variables, and number of non-zero terms, but have very different properties.

Another Classification

- A better way is to use the minimum number of operations needed to calculate the polynomial.
- This parameter captures differences (and similarities) in polynomials better:
  - Polynomials $\prod_{k=1}^{n} x_k$ and $\prod_{k=1}^{n}(1+x_k)$ can be computed in $n-1$ and $2n-1$ operations respectively.
  - Polynomials $\det_n$ requires $n^{O(1)}$ operations while per_n appears to require $2^{n^{\Omega(1)}}$ operations.
- This parameter is called arithmetic complexity of a polynomial.

Another Classification

- A better way is to use the minimum number of operations needed to calculate the polynomial.
- This parameter captures differences (and similarities) in polynomials better:
  - Polynomials $\prod_{k=1}^{n} x_k$ and $\prod_{k=1}^{n}(1 + x_k)$ can be computed in $n - 1$ and $2n - 1$ operations respectively.
  - Polynomials $\det_n$ requires $n^{O(1)}$ operations while per_n appears to require $2^{n^{\Omega(1)}}$ operations.
- This parameter is called arithmetic complexity of a polynomial.

Another Classification

- A better way is to use the minimum number of operations needed to calculate the polynomial.
- This parameter captures differences (and similarities) in polynomials better:
  - Polynomials $\prod_{k=1}^{n} x_k$ and $\prod_{k=1}^{n}(1 + x_k)$ can be computed in $n - 1$ and $2n - 1$ operations respectively.
  - Polynomials $\det_n$ requires $n^{O(1)}$ operations while per_n appears to require $2^{n^{\Omega(1)}}$ operations.
- This parameter is called arithmetic complexity of a polynomial.

Another Classification

- A better way is to use the minimum number of operations needed to calculate the polynomial.
- This parameter captures differences (and similarities) in polynomials better:
  - Polynomials $\prod_{k=1}^{n} x_k$ and $\prod_{k=1}^{n}(1 + x_k)$ can be computed in $n - 1$ and $2n - 1$ operations respectively.
  - Polynomials $\det_n$ requires $n^{O(1)}$ operations while per_n appears to require $2^{n^{\Omega(1)}}$ operations.
- This parameter is called arithmetic complexity of a polynomial.

Outline

Arithmetic Circuits

Arithmetic circuits over field F represent variables and a sequence of arithmetic operations over F such that:

- Variables are input to the circuit,
- Each operation is on variables or result of previous operations,
- The result of last operation is output of the circuit.

- Allowed operations are addition and multiplication.
- Use of constants from the field is also allowed.
- The output of any circuit is a polynomial in the input variables.

Arithmetic Circuits

Arithmetic circuits over field F represent variables and a sequence of arithmetic operations over F such that:

- Variables are input to the circuit,
- Each operation is on variables or result of previous operations,
- The result of last operation is output of the circuit.

- Allowed operations are addition and multiplication.
- Use of constants from the field is also allowed.
- The output of any circuit is a polynomial in the input variables.

Arithmetic Circuits

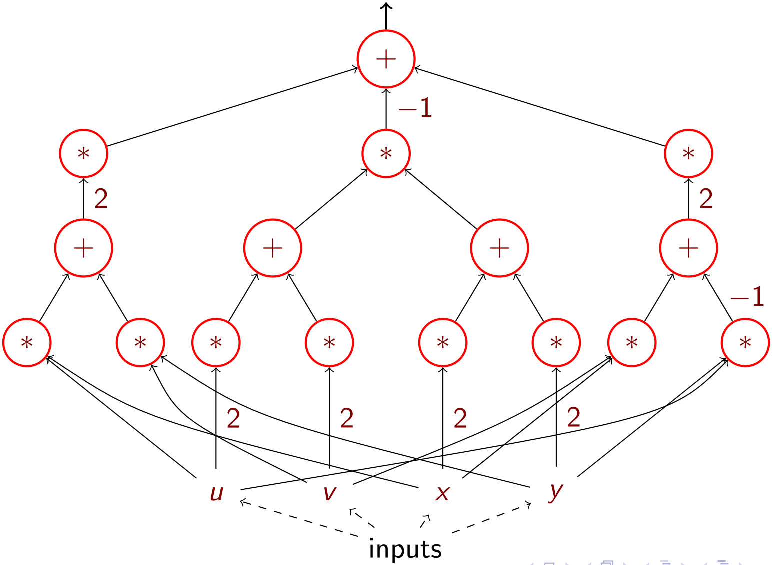
Arithmetic circuits over field F represent variables and a sequence of arithmetic operations over F such that:

- Variables are input to the circuit,
- Each operation is on variables or result of previous operations,
- The result of last operation is output of the circuit.

- Allowed operations are addition and multiplication.
- Use of constants from the field is also allowed.
- The output of any circuit is a polynomial in the input variables.

An Example

output $= (ux + vy)^2 + (vx - uy)^2 - (u^2 + v^2) \cdot (x^2 + y^2)$



Arithmetic Complexity

Crucial parameters associated with arithmetic circuits are:

- Size: equals the number of operations in the circuit.
- Depth: equals the length of the longest path from a variable to output of the circuit.

Arithmetic complexity $\mathcal{A}(P)$ of a polynomial P is the size of the smallest arithmetic circuit that outputs the polynomial.

Arithmetic Complexity

Crucial parameters associated with arithmetic circuits are:

- Size: equals the number of operations in the circuit.
- Depth: equals the length of the longest path from a variable to output of the circuit.

Arithmetic complexity $\mathcal{A}(P)$ of a polynomial P is the size of the smallest arithmetic circuit that outputs the polynomial.

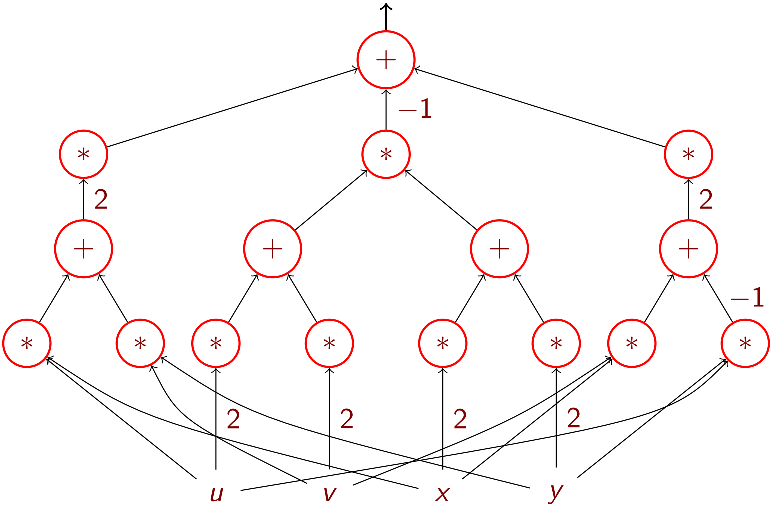
Arithmetic Complexity

Crucial parameters associated with arithmetic circuits are:

- Size: equals the number of operations in the circuit.
- Depth: equals the length of the longest path from a variable to output of the circuit.

Arithmetic complexity $\mathcal{A}(P)$ of a polynomial P is the size of the smallest arithmetic circuit that outputs the polynomial.

Circuit Parameters



SIZE $= 17$ DEPTH $= 4$ DEGREE $= 4$

Properties of Arithmetic Circuits

- Arithmetic circuits provide a compact way of representing polynomials.
- For example,

$$(1 + x_1) \cdot (1 + x_2) \cdot \cdots \cdot (1 + x_n)$$

  can be represented by an arithmetic circuit of size $2n - 1$ even though it has 2^n terms.
  - Arithmetic complexity of this polynomial is, therefore, at most $2n - 1$.
- Given a polynomial as arithmetic circuit, it can be evaluated at any point efficiently: in time proportional to the size of the circuit.

Note that we cannot say that the complexity of the polynomial is exactly $2n - 1$ as there may exist a better way of representing the polynomial.

Properties of Arithmetic Circuits

- Arithmetic circuits provide a compact way of representing polynomials.
- For example,

$$(1 + x_1) \cdot (1 + x_2) \cdot \cdots \cdot (1 + x_n)$$

  can be represented by an arithmetic circuit of size $2n - 1$ even though it has 2^n terms.
  - Arithmetic complexity of this polynomial is, therefore, at most $2n - 1$.
- Given a polynomial as arithmetic circuit, it can be evaluated at any point efficiently: in time proportional to the size of the circuit.

Note that we cannot say that the complexity of the polynomial is exactly $2n - 1$ as there may exist a better way of representing the polynomial.

Properties of Arithmetic Circuits

- Arithmetic circuits provide a compact way of representing polynomials.
- For example,

$$(1 + x_1) \cdot (1 + x_2) \cdot \cdots \cdot (1 + x_n)$$

  can be represented by an arithmetic circuit of size $2n - 1$ even though it has 2^n terms.
  - Arithmetic complexity of this polynomial is, therefore, at most $2n - 1$.
- Given a polynomial as arithmetic circuit, it can be evaluated at any point efficiently: in time proportional to the size of the circuit.

Note that we cannot say that the complexity of the polynomial is exactly $2n - 1$ as there may exist a better way of representing the polynomial.

Why No Division Operation?

Theorem

Given a circuit of size s, computing polynomial P of degree d, with addition, multiplication and division operations, all the division operations can be replaced with addition and multiplication operations at the cost of increasing the size to $(s+d)^{O(1)}$.

Arithmetic Complexity of Polynomial Families

- There are a number of interesting families of polynomials: Chebyshev polynomials ($T_d(x)$), Euler polynomials ($E_d(x)$), Determinant polynomials ($\det_n(x_{1,1}, \ldots, x_{n,n})$).
- Each family contains infinitely many polynomials of similar kind, with different degrees and, at times, different number of variables.
- To represent a family of polynomials, we use families of arithmetic circuits, one circuit for each polynomial in the family.
- The arithmetic complexity of such a family is measured as a function of n and d, the number of variables and degree of polynomials in the family.

Arithmetic Complexity of Polynomial Families

- There are a number of interesting families of polynomials: Chebyshev polynomials ($T_d(x)$), Euler polynomials ($E_d(x)$), Determinant polynomials ($\det_n(x_{1,1}, \ldots, x_{n,n})$).
- Each family contains infinitely many polynomials of similar kind, with different degrees and, at times, different number of variables.
- To represent a family of polynomials, we use families of arithmetic circuits, one circuit for each polynomial in the family.
- The arithmetic complexity of such a family is measured as a function of n and d, the number of variables and degree of polynomials in the family.

Arithmetic Complexity of Polynomial Families

- There are a number of interesting families of polynomials: Chebyshev polynomials ($T_d(x)$), Euler polynomials ($E_d(x)$), Determinant polynomials ($\det_n(x_{1,1}, \ldots, x_{n,n})$).
- Each family contains infinitely many polynomials of similar kind, with different degrees and, at times, different number of variables.
- To represent a family of polynomials, we use families of arithmetic circuits, one circuit for each polynomial in the family.
- The arithmetic complexity of such a family is measured as a function of n and d, the number of variables and degree of polynomials in the family.

Examples

- $\mathcal{A}(\prod_{k=1}^{n}(1+x_k)) = O(n)$.
- $\mathcal{A}(\det_n) = n^{O(1)}$. [Gaussian elimination]
- $\mathcal{A}(T_d) = \mathcal{A}(E_d) = O(d)$. [Polynomials are of degree $O(d)$]

Examples

- $\mathcal{A}(\prod_{k=1}^{n}(1+x_k)) = O(n)$.
- $\mathcal{A}(\det_n) = n^{O(1)}$. [Gaussian elimination]
- $\mathcal{A}(T_d) = \mathcal{A}(E_d) = O(d)$. [Polynomials are of degree $O(d)$]

Examples

- $\mathcal{A}(\prod_{k=1}^{n}(1+x_k)) = O(n)$.
- $\mathcal{A}(\mathrm{det}_n) = n^{O(1)}$. [Gaussian elimination]
- $\mathcal{A}(T_d) = \mathcal{A}(E_d) = O(d)$. [Polynomials are of degree $O(d)$]

Bounds on Arithmetic Complexity

Upper Bound

Let P be a polynomial over n variables and degree d. Then

$$\mathcal{A}(P) \leq dn(d+1)^n \leq 2^{(n+\log d)^2}.$$

Proof.

Write the polynomial as a sum of $(d+1)^n$ terms. Each terms requires at most dn multiplications.

Bounds on Arithmetic Complexity

Lower Bound

Let P be a polynomial over n variables and degree d. Then

$$\mathcal{A}(P) = \Omega(n + \log d).$$

Proof.

$\mathcal{A}(P) \geq n - 1$ since each variable participates in at least one operation.

$\mathcal{A} \geq \log d$ since each multiplication at most doubles the degree.

Classifying Arithmetic Complexity of Polynomial Families

- Polynomial family $\{P_{n,d}\}$ has low arithmetic complexity if $\mathcal{A}(P_{n,d}) = (n + \log d)^{O(1)}$:
  - The complexity is close to the minimum possible.
  - Intuitively, such polynomials are "simple".
  - Examples: $\{\prod_{k=1}^{n}(1 + x_k)\}$ and $\{\det_n\}$.
- Polynomial family $\{P_{n,d}\}$ has high arithmetic complexity if $\mathcal{A}(P_{n,d}) = 2^{(n+\log d)^{\Omega(1)}}$:
  - The complexity is close to the maximum possible.
  - Intuitively, such polynomials are "difficult".
  - Examples: Conjecturally $\{\mathrm{per}_n\}$ and $\{E_d(x)\}$.
- We now define two classes of polynomial families: one capturing families of low complexity, and other capturing most of the interesting polynomials (even of high complexity).

Classifying Arithmetic Complexity of Polynomial Families

- Polynomial family $\{P_{n,d}\}$ has low arithmetic complexity if $\mathcal{A}(P_{n,d}) = (n + \log d)^{O(1)}$:
  - The complexity is close to the minimum possible.
  - Intuitively, such polynomials are "simple".
  - Examples: $\{\prod_{k=1}^{n}(1 + x_k)\}$ and $\{\det_n\}$.
- Polynomial family $\{P_{n,d}\}$ has high arithmetic complexity if $\mathcal{A}(P_{n,d}) = 2^{(n+\log d)^{\Omega(1)}}$:
  - The complexity is close to the maximum possible.
  - Intuitively, such polynomials are "difficult".
  - Examples: Conjecturally $\{\mathrm{per}_n\}$ and $\{E_d(x)\}$.
- We now define two classes of polynomial families: one capturing families of low complexity, and other capturing most of the interesting polynomials (even of high complexity).

Classifying Arithmetic Complexity of Polynomial Families

- Polynomial family $\{P_{n,d}\}$ has low arithmetic complexity if $\mathcal{A}(P_{n,d}) = (n + \log d)^{O(1)}$:
  - The complexity is close to the minimum possible.
  - Intuitively, such polynomials are "simple".
  - Examples: $\{\prod_{k=1}^{n}(1 + x_k)\}$ and $\{\det_n\}$.
- Polynomial family $\{P_{n,d}\}$ has high arithmetic complexity if $\mathcal{A}(P_{n,d}) = 2^{(n+\log d)^{\Omega(1)}}$:
  - The complexity is close to the maximum possible.
  - Intuitively, such polynomials are "difficult".
  - Examples: Conjecturally $\{\mathrm{per}_n\}$ and $\{E_d(x)\}$.
- We now define two classes of polynomial families: one capturing families of low complexity, and other capturing most of the interesting polynomials (even of high complexity).

Classifying Arithmetic Complexity of Polynomial Families

- Polynomial family $\{P_{n,d}\}$ has low arithmetic complexity if $\mathcal{A}(P_{n,d}) = (n + \log d)^{O(1)}$:
  - The complexity is close to the minimum possible.
  - Intuitively, such polynomials are "simple".
  - Examples: $\{\prod_{k=1}^{n}(1 + x_k)\}$ and $\{\det_n\}$.
- Polynomial family $\{P_{n,d}\}$ has high arithmetic complexity if $\mathcal{A}(P_{n,d}) = 2^{(n+\log d)^{\Omega(1)}}$:
  - The complexity is close to the maximum possible.
  - Intuitively, such polynomials are "difficult".
  - Examples: Conjecturally $\{\mathrm{per}_n\}$ and $\{E_d(x)\}$.
- We now define two classes of polynomial families: one capturing families of low complexity, and other capturing most of the interesting polynomials (even of high complexity).

Outline

Capturing Low Complexity Families

The Class VP

Polynomial family $\{P_{n,d}\}$ is in VP if $\mathcal{A}(P_{n,d}) \leq (n + \log d)^{O(1)}$ for all n and d.

- The polynomial families in VP are precisely the low arithmetic complexity families.
- Each of these families has a number of nice properties due to being a 'simple' family.

Capturing Low Complexity Families

The Class VP

Polynomial family $\{P_{n,d}\}$ is in VP if $\mathcal{A}(P_{n,d}) \leq (n + \log d)^{O(1)}$ for all n and d.

- The polynomial families in VP are precisely the low arithmetic complexity families.
- Each of these families has a number of nice properties due to being a 'simple' family.

Capturing Low Complexity Families

The Class VP

Polynomial family $\{P_{n,d}\}$ is in VP if $\mathcal{A}(P_{n,d}) \leq (n + \log d)^{O(1)}$ for all n and d.

- The polynomial families in VP are precisely the low arithmetic complexity families.
- Each of these families has a number of nice properties due to being a 'simple' family.

Examples: Polynomial Families in VP

- $\{\det_n\}$
- Elementary symmetric polynomials:

$$S_{n,d} = \sum_{1 \leq k_1 \neq k_2 \neq \cdots \neq k_d \leq n} \prod_{i=1}^{d} x_{k_i}.$$

- Family $\{T_d\}$, exploiting the following property:

$$T_d(x) = \frac{(x - \sqrt{x^2 - 1})^d + (x + \sqrt{x^2 - 1})^d}{2}.$$

Examples: Polynomial Families in VP

- $\{\det_n\}$
- Elementary symmetric polynomials:

$$S_{n,d} = \sum_{1 \le k_1 \ne k_2 \ne \cdots \ne k_d \le n} \prod_{i=1}^{d} x_{k_i}.$$

- Family $\{T_d\}$, exploiting the following property:

$$T_d(x) = \frac{(x - \sqrt{x^2 - 1})^d + (x + \sqrt{x^2 - 1})^d}{2}.$$

Examples: Polynomial Families in VP

- $\{\det_n\}$
- Elementary symmetric polynomials:

$$S_{n,d} = \sum_{1 \leq k_1 \neq k_2 \neq \cdots \neq k_d \leq n} \prod_{i=1}^{d} x_{k_i}.$$

- Family $\{T_d\}$, exploiting the following property:

$$T_d(x) = \frac{(x - \sqrt{x^2 - 1})^d + (x + \sqrt{x^2 - 1})^d}{2}.$$

Capturing Interesting Families

- The class VP does not appear to contain several interesting families of polynomials, e.g., $\{\mathrm{per}_n\}$, $\{E_d\}$.
- To capture these polynomials, we identify a common property: each can be written as a large sum of simple monomials.

Capturing Interesting Families

- The class VP does not appear to contain several interesting families of polynomials, e.g., $\{\mathrm{per}_n\}$, $\{E_d\}$.
- To capture these polynomials, we identify a common property: each can be written as a large sum of simple monomials.

Capturing Interesting Families

$$\mathrm{per}_n = \sum_{\pi \in S_n} \prod_{i=1}^{n} x_{i,\pi(i)}$$

- It is a large sum $(= n!)$ of monomials, each of which is very simple: $\prod_{i=1}^{n} x_{i,\pi(i)}$ for some $\pi \in S_n$.

$$E_d(x) = \sum_{k=0}^{d} c_k x^k$$

- It is a large sum $(= d+1)$ of monomials, each of which is $c_k x^k$ for some $0 \leq k \leq n$.
- The constants c_k can be computed in $\mathrm{P}^{\#\mathrm{P}}$, and so the monomials above are 'somewhat' simple.

Capturing Interesting Families

$$\text{per}_n = \sum_{\pi \in S_n} \prod_{i=1}^{n} x_{i,\pi(i)}$$

- It is a large sum ($= n!$) of monomials, each of which is very simple: $\prod_{i=1}^{n} x_{i,\pi(i)}$ for some $\pi \in S_n$.

$$E_d(x) = \sum_{k=0}^{d} c_k x^k$$

- It is a large sum ($= d + 1$) of monomials, each of which is $c_k x^k$ for some $0 \leq k \leq n$.
- The constants c_k can be computed in $\mathsf{P}^{\#\mathsf{P}}$, and so the monomials above are 'somewhat' simple.

Capturing Interesting Families

The Class VNP

Polynomial family $\{Q_{n,d}\}$ is in VNP if there exists a family $\{P_{n,d}\} \in$ VP such that for every n and d:

$$Q_{n,d}(x_1, \ldots, x_n) = \sum_{y_1=0}^{1} \cdots \sum_{y_n=0}^{1} P_{2n,d}(x_1, \ldots, x_n, y_1, \ldots, y_n).$$

VP and VNP are algebraic analog of the classes P and NP.

Capturing Interesting Families

The Class VNP

Polynomial family $\{Q_{n,d}\}$ is in VNP if there exists a family $\{P_{n,d}\} \in$ VP such that for every n and d:

$$Q_{n,d}(x_1, \ldots, x_n) = \sum_{y_1=0}^{1} \cdots \sum_{y_n=0}^{1} P_{2n,d}(x_1, \ldots, x_n, y_1, \ldots, y_n).$$

VP and VNP are algebraic analog of the classes P and NP.

Capturing Interesting Families

The Class VNP

Polynomial family $\{Q_{n,d}\}$ is in VNP if there exists a family $\{P_{n,d}\} \in$ VP such that for every n and d:

$$Q_{n,d}(x_1, \ldots, x_n) = \sum_{y_1=0}^{1} \cdots \sum_{y_n=0}^{1} P_{2n,d}(x_1, \ldots, x_n, y_1, \ldots, y_n).$$

VP and VNP are algebraic analog of the classes P and NP.

Examples: Polynomial Families in VNP

- All polynomial families in VP
- $\{\mathrm{per}_n\}$
- Jones polynomials: representing invariants of knots
- Tutte polynomials:

$$T_G(x,y) = \sum_{A \subseteq E} (x-1)^{k(A)-k(E)}(y-1)^{k(A)+|A|-|V|}$$

  where $G=(V,E)$ is an undirected graph and $k(A)$ is the number of connected components in the subgraph (V,A).
- Univariate polynomials whose coefficients can be computed in $\mathrm{P}^{\#\mathrm{P}}$ are all in $\mathrm{VNP}^{\#\mathrm{P}}$.

Examples: Polynomial Families in VNP

- All polynomial families in VP
- $\{\mathrm{per}_n\}$
- Jones polynomials: representing invariants of knots
- Tutte polynomials:

$$T_G(x,y) = \sum_{A \subseteq E} (x-1)^{k(A)-k(E)}(y-1)^{k(A)+|A|-|V|}$$

  where $G = (V, E)$ is an undirected graph and $k(A)$ is the number of connected components in the subgraph (V, A).
- Univariate polynomials whose coefficients can be computed in $\mathrm{P}^{\#\mathrm{P}}$ are all in $\mathrm{VNP}^{\#\mathrm{P}}$.

Examples: Polynomial Families in VNP

- All polynomial families in VP
- $\{\text{per}_n\}$
- Jones polynomials: representing invariants of knots
- Tutte polynomials:

$$T_G(x, y) = \sum_{A \subseteq E} (x-1)^{k(A)-k(E)} (y-1)^{k(A)+|A|-|V|}$$

  where $G = (V, E)$ is an undirected graph and $k(A)$ is the number of connected components in the subgraph (V, A).
- Univariate polynomials whose coefficients can be computed in $\mathrm{P}^{\#\mathrm{P}}$ are all in $\mathrm{VNP}^{\#\mathrm{P}}$.

Examples: Polynomial Families in VNP

- All polynomial families in VP
- $\{\mathrm{per}_n\}$
- Jones polynomials: representing invariants of knots
- Tutte polynomials:

$$T_G(x,y) = \sum_{A \subseteq E} (x-1)^{k(A)-k(E)} (y-1)^{k(A)+|A|-|V|}$$

where $G = (V, E)$ is an undirected graph and $k(A)$ is the number of connected components in the subgraph (V, A).

- Univariate polynomials whose coefficients can be computed in $\mathrm{P}^{\#\mathrm{P}}$ are all in $\mathrm{VNP}^{\#\mathrm{P}}$.

Examples: Polynomial Families in VNP

- All polynomial families in VP
- $\{\mathrm{per}_n\}$
- Jones polynomials: representing invariants of knots
- Tutte polynomials:

$$T_G(x, y) = \sum_{A \subseteq E} (x-1)^{k(A)-k(E)} (y-1)^{k(A)+|A|-|V|}$$

  where $G = (V, E)$ is an undirected graph and $k(A)$ is the number of connected components in the subgraph (V, A).
- Univariate polynomials whose coefficients can be computed in $\mathrm{P}^{\#\mathrm{P}}$ are all in $\mathrm{VNP}^{\#\mathrm{P}}$.

Permanent Family and VNP

Theorem [Valient 1979]

Family $\{\mathrm{per}_n\}$ is complete for VNP: for every polynomial family $\{Q_{n,d}\}$ in VNP, for every n and d, $Q_{n,d}$ can be expressed as permanent of a $(n + \log d)^{O(1)}$-size matrix.

- This implies that $\{\mathrm{per}_n\}$ is the "hardest" family in VNP.

Permanent Family and VNP

Theorem [Valiant 1979]

Family $\{\text{per}_n\}$ is complete for VNP: for every polynomial family $\{Q_{n,d}\}$ in VNP, for every n and d, $Q_{n,d}$ can be expressed as permanent of a $(n + \log d)^{O(1)}$-size matrix.

- This implies that $\{\text{per}_n\}$ is the "hardest" family in VNP.

Is VP $\neq$ VNP?

- If VP $\neq$ VNP, it follows that several families (Permanent, Jones polynomial, Tutte polynomial etc) do not have low complexity.
- It is believed that VP $\neq$ VNP, but no proof is known.
- Over the years, it has become one of the most active areas of research in complexity theory.
- Polynomial Identity Testing provides an approach to solve this problem.

Is VP $\neq$ VNP?

- If VP $\neq$ VNP, it follows that several families (Permanent, Jones polynomial, Tutte polynomial etc) do not have low complexity.
- It is believed that VP $\neq$ VNP, but no proof is known.
- Over the years, it has become one of the most active areas of research in complexity theory.
- Polynomial Identity Testing provides an approach to solve this problem.

Is VP $\neq$ VNP?

- If VP $\neq$ VNP, it follows that several families (Permanent, Jones polynomial, Tutte polynomial etc) do not have low complexity.
- It is believed that VP $\neq$ VNP, but no proof is known.
- Over the years, it has become one of the most active areas of research in complexity theory.
- Polynomial Identity Testing provides an approach to solve this problem.

Outline

Polynomial Identity Testing

Definition

Given an arithmetic circuit of size s, test if the polynomial computed by the circuit is non-zero.

A Simpler Version

Given an arithmetic circuit of size s computing a polynomial of degree $\leq s$, test if the polynomial computed by the circuit is non-zero.

- The problems are referred as PIT and LPIT in short.

Polynomial Identity Testing

Definition

Given an arithmetic circuit of size s, test if the polynomial computed by the circuit is non-zero.

A Simpler Version

Given an arithmetic circuit of size s computing a polynomial of degree $\leq s$, test if the polynomial computed by the circuit is non-zero.

- The problems are referred as PIT and LPIT in short.

Polynomial Identity Testing

Definition

Given an arithmetic circuit of size s, test if the polynomial computed by the circuit is non-zero.

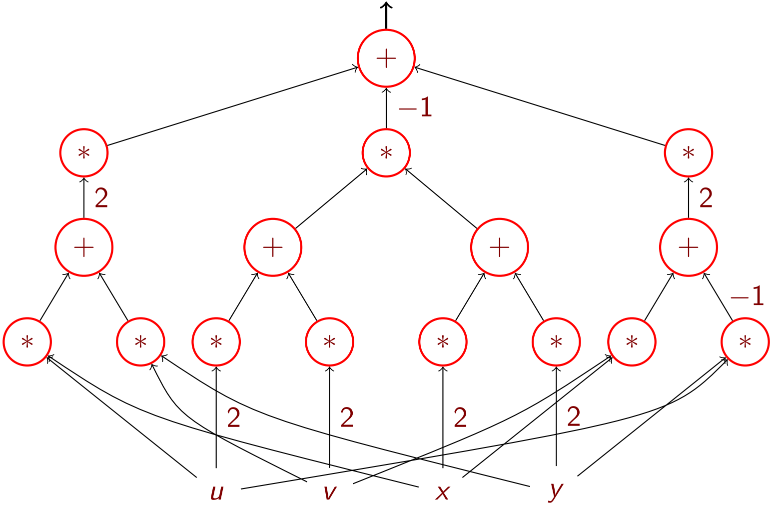
A Simpler Version

Given an arithmetic circuit of size s computing a polynomial of degree $\leq s$, test if the polynomial computed by the circuit is non-zero.

- The problems are referred as PIT and LPIT in short.

An Example

Is $(ux+vy)^2 + (vx-uy)^2 - (u^2+v^2)\cdot(x^2+y^2) \neq 0$? [NO!]



Complexity of PIT

A number of randomized polynomial time algorithms are known for the problem.

- The simplest one is by [Schwartz, Zippel 1979]: Substitute random values from a small subset of $\mathbb{Q}$ for each variable, evaluate the circuit, and output NON-ZERO iff the result is a non-zero number.
- Others are [Chen-Kao 1997], [Lewis-Vadhan 1998], [A-Biswas 1999], ...

Complexity of PIT

A number of randomized polynomial time algorithms are known for the problem.

- The simplest one is by [Schwartz, Zippel 1979]: Substitute random values from a small subset of $\mathbb{Q}$ for each variable, evaluate the circuit, and output NON-ZERO iff the result is a non-zero number.
- Others are [Chen-Kao 1997], [Lewis-Vadhan 1998], [A-Biswas 1999], ...

Complexity of PIT

A number of randomized polynomial time algorithms are known for the problem.

- The simplest one is by [Schwartz, Zippel 1979]: Substitute random values from a small subset of $\mathbb{Q}$ for each variable, evaluate the circuit, and output NON-ZERO iff the result is a non-zero number.
- Others are [Chen-Kao 1997], [Lewis-Vadhan 1998], [A-Biswas 1999], ...

Deterministic Algorithm for PIT

Open Question

Is there a deterministic polynomial time algorithm for PIT?

- Long-standing open problem.
- Also open for LPIT.
- A positive answer also yields a lower bound!

Deterministic Algorithm for PIT

Open Question

Is there a deterministic polynomial time algorithm for PIT?

- Long-standing open problem.
- Also open for LPIT.
- A positive answer also yields a lower bound!

Deterministic Algorithm for PIT

Open Question

Is there a deterministic polynomial time algorithm for PIT?

- Long-standing open problem.
- Also open for LPIT.
- A positive answer also yields a lower bound!

Outline

Previous Century

1970s : Definition and first randomized algorithm [Schwartz-Zippel] for PIT.

1980s : Applications, e.g., multi-set equality, graph matching.

1990s :

- More applications, e.g., IP = PSPACE, PCP Theorem.
- More randomized algorithms [Chen-Kao 1997] (for LPIT), [Lewis-Vadhan 1998] (for LPIT), [A-Biswas 1999] (for PIT).

Previous Century

1970s : Definition and first randomized algorithm [Schwartz-Zippel] for PIT.

1980s : Applications, e.g., multi-set equality, graph matching.

1990s :

- More applications, e.g., IP = PSPACE, PCP Theorem.
- More randomized algorithms [Chen-Kao 1997] (for LPIT), [Lewis-Vadhan 1998] (for LPIT), [A-Biswas 1999] (for PIT).

Previous Century

1970s : Definition and first randomized algorithm [Schwartz-Zippel] for PIT.

1980s : Applications, e.g., multi-set equality, graph matching.

1990s :

- More applications, e.g., IP $=$ PSPACE, PCP Theorem.
- More randomized algorithms [Chen-Kao 1997] (for LPIT), [Lewis-Vadhan 1998] (for LPIT), [A-Biswas 1999] (for PIT).

2001-04

- Yet another randomized algorithm [Klivans-Spielman 2001] (for PIT).
- Deterministic algorithm for a special class of identities on single variable: primality test.
- Connection of deterministic algorithm with lower bounds [Kabanets-Impagliazzo 2003]!

THEOREM

If there exists a deterministic polynomial time algorithm for LPIT then either $NEXP \notin P/poly$ or $VP \neq VNP$.

2001-04

- Yet another randomized algorithm [Klivans-Spielman 2001] (for PIT).
- Deterministic algorithm for a special class of identities on single variable: primality test.
- Connection of deterministic algorithm with lower bounds [Kabanets-Impagliazzo 2003]!

Theorem

If there exists a deterministic polynomial time algorithm for LPIT then either $NEXP \notin P/poly$ or $VP \neq VNP$.

2001-04

- Yet another randomized algorithm [Klivans-Spielman 2001] (for PIT).
- Deterministic algorithm for a special class of identities on single variable: primality test.
- Connection of deterministic algorithm with lower bounds [Kabanets-Impagliazzo 2003]!

Theorem

If there exists a deterministic polynomial time algorithm for LPIT then either $NEXP \notin P/poly$ or $VP \neq VNP$.

2005-08

- Deterministic polynomial time algorithm for non-commutative LPIT for formulas [Raz-Shpilka 2005].
- Deterministic $2^{(\log d)^{k^2}}$ time algorithm for 3-PIT with top fanin k [Dvir-Shpilka 2005].

c-PIT

c-PIT is the restriction of PIT to depth c circuits in which unbounded fanin gates are allowed and top gate is a $+$.

2005-08

- Deterministic polynomial time algorithm for non-commutative LPIT for formulas [Raz-Shpilka 2005].
- Deterministic $2^{(\log d)^{k^2}}$ time algorithm for 3-PIT with top fanin k [Dvir-Shpilka 2005].

c-PIT

c-PIT is the restriction of PIT to depth c circuits in which unbounded fanin gates are allowed and top gate is a $+$.

2005-08

- Another connection with lower bounds [A 2005].

Theorem

If there exist deterministic polynomial time black-box algorithm for LPIT then there exists a polynomial family, computable in exponential time, that is not in VP.

A black-box derandomization of PIT is a deterministic algorithm that can feed inputs to the circuit and see the output, but does not have access to the structure of the circuit except the knowledge of its size, depth, and degree.

2005-08

- Another connection with lower bounds [A 2005].

Theorem

If there exist deterministic polynomial time black-box algorithm for LPIT then there exists a polynomial family, computable in exponential time, that is not in VP.

A black-box derandomization of PIT is a deterministic algorithm that can feed inputs to the circuit and see the output, but does not have access to the structure of the circuit except the knowledge of its size, depth, and degree.

2005-08

- Deterministic $d^{O(k)}$ time algorithm for 3-PIT with top fanin k [Kayal-Saxena 2006].
- $d^{O\log^k d}$ time black-box algorithm for 3-PIT with top fanin k [Karnin-Shpilka 2008].
- Connection of 4-PIT with lower bounds [A-Vinay 2008].

Theorem

If there exist deterministic polynomial time black-box algorithm for 4-PIT then there exists a polynomial family, computable in exponential time, that is not in VP.

2005-08

- Deterministic $d^{O(k)}$ time algorithm for 3-PIT with top fanin k [Kayal-Saxena 2006].
- $d^{O\log^k d}$ time black-box algorithm for 3-PIT with top fanin k [Karnin-Shpilka 2008].
- Connection of 4-PIT with lower bounds [A-Vinay 2008].

Theorem

If there exist deterministic polynomial time black-box algorithm for 4-PIT then there exists a polynomial family, computable in exponential time, that is not in VP.

2005-08

- Deterministic $d^{O(k)}$ time algorithm for 3-PIT with top fanin k [Kayal-Saxena 2006].
- $d^{O \log^k d}$ time black-box algorithm for 3-PIT with top fanin k [Karnin-Shpilka 2008].
- Connection of 4-PIT with lower bounds [A-Vinay 2008].

Theorem

If there exist deterministic polynomial time black-box algorithm for 4-PIT then there exists a polynomial family, computable in exponential time, that is not in VP.

2009-11

- $d^{k^3 \log d}$ time black-box algorithm for 3-PIT with top fanin k [Saxena-Seshadri 2009].
- d^{k^k} time black-box algorithm for 3-PIT with top fanin k over characteristic zero fields [Kayal-Saraf 2009].
- $d^{O(k)}$ time black-box algorithm for 3-PIT with top fanin k [Saxena-Seshadri 2011].

2009-11

- $d^{k^3 \log d}$ time black-box algorithm for 3-PIT with top fanin k [Saxena-Seshadri 2009].
- d^{k^k} time black-box algorithm for 3-PIT with top fanin k over characteristic zero fields [Kayal-Saraf 2009].
- $d^{O(k)}$ time black-box algorithm for 3-PIT with top fanin k [Saxena-Seshadri 2011].

2009-11

- $d^{k^3 \log d}$ time black-box algorithm for 3-PIT with top fanin k [Saxena-Seshadri 2009].
- d^{k^k} time black-box algorithm for 3-PIT with top fanin k over characteristic zero fields [Kayal-Saraf 2009].
- $d^{O(k)}$ time black-box algorithm for 3-PIT with top fanin k [Saxena-Seshadri 2011].

2009-11

- s^{k^3} time black-box algorithm for 4-PIT on multilinear circuits with top fanin k [Saraf-Volkovich].
- Connecting derandomization to VP $\neq$ VNP [A 2011].

DEFINITION

A multilinear circuit is one in which every gate computes a multilinear polynomial.

THEOREM

If a special black-box polynomial time algorithm, based in Euler polynomials, solves 4-PIT then VP $\neq$ VNP.

2009-11

- s^{k^3} time black-box algorithm for 4-PIT on multilinear circuits with top fanin k [Saraf-Volkovich].
- Connecting derandomization to VP $\neq$ VNP [A 2011].

Definition

A multilinear circuit is one in which every gate computes a multilinear polynomial.

Theorem

If a special black-box polynomial time algorithm, based in Euler polynomials, solves 4-PIT then VP $\neq$ VNP.

2012

- s^{k^2} time black-box algorithm for 4-PIT in which every variables occurs in at most k level two polynomials [A-Saha-Saptharishi-Saxena 2012].
- $s^{O(\log s)}$ time black-box algorithm for 4-PIT on set-multilinear circuits [A-Saha-Saxena 2012].

Definition

A set-multilinear circuit of depth four is one in which every level three gate is a product over the same disjoint set of variables.

2012

- s^{k^2} time black-box algorithm for 4-PIT in which every variables occurs in at most k level two polynomials [A-Saha-Saptharishi-Saxena 2012].
- $s^{O(\log s)}$ time black-box algorithm for 4-PIT on set-multilinear circuits [A-Saha-Saxena 2012].

Definition

A set-multilinear circuit of depth four is one in which every level three gate is a product over the same disjoint set of variables.

THANK YOU!