The Arithmetic Complexity of Euler Function

Manindra Agrawal

IIT Kanpur

Computer Science Symposium in Russia, June 2011

Outline

Euler Function

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

Defined by Leonhard Euler.

Relation to Partition Numbers

Let p_m be the number of partitions of m. Then

$$\frac{1}{E(x)} = \sum_{m \geq 0} p_m x^m.$$

Proof. Note that

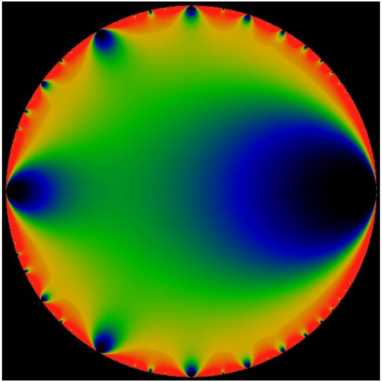
$$\frac{1}{E(x)} = \frac{1}{\prod_{k>0}(1 - x^k)} = \prod_{k>0}(\sum_{t \geq 0} x^{kt}).$$

Euler Identity

$$E(x) = \sum_{m=-\infty}^{\infty} (-1)^m x^{(3m^2-m)/2}.$$

Proof. Set up an involution between terms of same degree and opposite signs. Only a few survive.

Shape Over Complex Plane



- Undefined outside unit disk.
- Zero at unit circle.
- Bounded inside the unit disk:
  - Red represents value 4
  - Black represents value 0

Image created by Linas Vepstas (linas@linas.org) and released under the Gnu Free Documentation License (GFDL). Borrowed from http://en.wikipedia.org/wiki/File:Q-euler.jpeg.

Capturing Symmetries

Dedekind Eta Function

$$\eta(z) = e^{\frac{\pi i z}{12}} E(e^{2\pi i z}).$$

$\eta(z)$ is defined on the upper half of the complex plane and satisfies many interesting properties:

- $\eta(z+1) = e^{\frac{\pi i}{12}} \eta(z)$.
- $\eta(-\frac{1}{z}) = \sqrt{-iz}\,\eta(z)$.

Proof. First part is trivial. Second part requires non-trivial complex analysis.

Permanent Polynomial

- For any $n > 0$, let $X = [x_{i,j}]$ be a $n \times n$ matrix with variable elements.
- Then permanent polynomial of degree n is the permanent of X:

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

- It is believed to be hard to compute.

Permanent Polynomial

- For any $n > 0$, let $X = [x_{i,j}]$ be a $n \times n$ matrix with variable elements.
- Then permanent polynomial of degree n is the permanent of X:

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

- It is believed to be hard to compute.

Permanent Polynomial

- For any $n > 0$, let $X = [x_{i,j}]$ be a $n \times n$ matrix with variable elements.
- Then permanent polynomial of degree n is the permanent of X:

$$\text{per}_{\,n}(\bar{x}) = \sum_{\sigma \in S_n} \prod_{i=1}^{n} x_{i,\sigma(i)}.$$

- It is believed to be hard to compute.

Outline

Two Families of Polynomials

- Let
$$E_{\Sigma,n}(x) = \sum_{k=-n}^{n} (-1)^k x^{(3k^2-k)/2}$$
and
$$E_{\Pi,n} = \prod_{k=1}^{n} (1 - x^k).$$
- We have:
  - $E(x) = \lim_{n \mapsto \infty} E_{\Sigma,n}(x) = \lim_{n \mapsto \infty} E_{\Pi,n}(x)$.
  - $E_{\Sigma,n}(x)$ is a polynomial of degree $\frac{1}{2}(3n^2 + n)$ and $E_{\Pi,n}(x)$ is a polynomial of degree $\frac{1}{2}(n^2 + n)$.
- A circuit family computing $E_{\Sigma,n}(x)$ or $E_{\Pi,n}(x)$ can be viewed as computing $E(x)$.
- We will consider arithmetic circuit families for computing $E_{\Sigma,n}(x)$ and $E_{\Pi,n}(x)$.

Two Families of Polynomials

- Let

$$E_{\Sigma,n}(x) = \sum_{k=-n}^{n} (-1)^k x^{(3k^2-k)/2}$$

and

$$E_{\Pi,n} = \prod_{k=1}^{n} (1 - x^k).$$

- We have:
  - $E(x) = \lim_{n \mapsto \infty} E_{\Sigma,n}(x) = \lim_{n \mapsto \infty} E_{\Pi,n}(x)$.
  - $E_{\Sigma,n}(x)$ is a polynomial of degree $\frac{1}{2}(3n^2 + n)$ and $E_{\Pi,n}(x)$ is a polynomial of degree $\frac{1}{2}(n^2 + n)$.
- A circuit family computing $E_{\Sigma,n}(x)$ or $E_{\Pi,n}(x)$ can be viewed as computing $E(x)$.
- We will consider arithmetic circuit families for computing $E_{\Sigma,n}(x)$ and $E_{\Pi,n}(x)$.

Two Families of Polynomials

- Let
$$E_{\Sigma,n}(x) = \sum_{k=-n}^{n} (-1)^k x^{(3k^2-k)/2}$$
and
$$E_{\Pi,n} = \prod_{k=1}^{n} (1 - x^k).$$

- We have:
  - $E(x) = \lim_{n \mapsto \infty} E_{\Sigma,n}(x) = \lim_{n \mapsto \infty} E_{\Pi,n}(x)$.
  - $E_{\Sigma,n}(x)$ is a polynomial of degree $\frac{1}{2}(3n^2 + n)$ and $E_{\Pi,n}(x)$ is a polynomial of degree $\frac{1}{2}(n^2 + n)$.
- A circuit family computing $E_{\Sigma,n}(x)$ or $E_{\Pi,n}(x)$ can be viewed as computing $E(x)$.
- We will consider arithmetic circuit families for computing $E_{\Sigma,n}(x)$ and $E_{\Pi,n}(x)$.

Arithmetic Circuits for Univariate Polynomials

- A circuit computing polynomial $P(x)$ over field F takes as input x and -1; and outputs $P(x)$.
- It is allowed to use addition and multiplication gates of arbitrary fanin over F.
- Size of a circuit is the number of wires in it.
- A depth two circuit family of size $O(n^2)$ can compute both $E_{\Sigma,n}(x)$ and $E_{\Pi,n}(x)$ over any field as they are polynomials of degree $O(n^2)$.
- A depth three circuit family of size $O(n)$ can compute $E_{\Pi,n}(x)$ over any field: follows from definition.

Arithmetic Circuits for Univariate Polynomials

- A circuit computing polynomial $P(x)$ over field F takes as input x and -1; and outputs $P(x)$.
- It is allowed to use addition and multiplication gates of arbitrary fanin over F.
- Size of a circuit is the number of wires in it.
- A depth two circuit family of size $O(n^2)$ can compute both $E_{\Sigma,n}(x)$ and $E_{\Pi,n}(x)$ over any field as they are polynomials of degree $O(n^2)$.
- A depth three circuit family of size $O(n)$ can compute $E_{\Pi,n}(x)$ over any field: follows from definition.

Arithmetic Circuits for Univariate Polynomials

- A circuit computing polynomial $P(x)$ over field F takes as input x and -1; and outputs $P(x)$.
- It is allowed to use addition and multiplication gates of arbitrary fanin over F.
- Size of a circuit is the number of wires in it.
- A depth two circuit family of size $O(n^2)$ can compute both $E_{\Sigma,n}(x)$ and $E_{\Pi,n}(x)$ over any field as they are polynomials of degree $O(n^2)$.
- A depth three circuit family of size $O(n)$ can compute $E_{\Pi,n}(x)$ over any field: follows from definition.

Circuits for $E(x)$

- Can a higher depth circuit do significantly better?
- For some other polynomials, we can do substantially better. For example,

$$\prod_{j=0}^{\log n-1}(1+x^{2^j}) = \sum_{i=0}^{n-1} x^i$$

can be computed by a depth three circuit of size $O(\log n)$.
- However, it is not clear how to compute $E(x)$ with $n^{o(1)}$ sized circuits.

Circuits for $E(x)$

- Can a higher depth circuit do significantly better?
- For some other polynomials, we can do substantially better. For example,

$$\prod_{j=0}^{\log n-1}(1+x^{2^j}) = \sum_{i=0}^{n-1} x^i$$

can be computed by a depth three circuit of size $O(\log n)$.

- However, it is not clear how to compute $E(x)$ with $n^{o(1)}$ sized circuits.

Circuits for $E(x)$

- Can a higher depth circuit do significantly better?
- For some other polynomials, we can do substantially better. For example,

$$\prod_{j=0}^{\log n-1}(1+x^{2^j}) = \sum_{i=0}^{n-1} x^i$$

  can be computed by a depth three circuit of size $O(\log n)$.
- However, it is not clear how to compute $E(x)$ with $n^{o(1)}$ sized circuits.

The Main Theorems

Theorem (First Theorem)

Suppose every circuit family computing $E_{\Sigma,n}(x)$ over F, char(F) > 2, has size $s(n^{\Omega(1)})$ for some $s(m) \geq (\log m)^2$. Then permanent polynomial family requires arithmetic circuits of size $s(2^{\Omega(n)})$ over F.

Theorem (Second Theorem)

Suppose every circuit family computing $E_{\Pi,n}(x)$ over F, char(F) > 2, has size $s(2^{\Omega(s(n^{O(1)}))})$ for some $s(m) \geq (\log m)^2$. Then permanent polynomial family requires arithmetic circuits of size $s(2^{\Omega(n)})$ over $\mathbb{Z}$.

A weaker version of second theorem was recently shown by Pascal Koiran.

The Main Theorems

Theorem (First Theorem)

Suppose every circuit family computing $E_{\Sigma,n}(x)$ over F, char(F) > 2, has size $s(n^{\Omega(1)})$ for some $s(m) \geq (\log m)^2$. Then permanent polynomial family requires arithmetic circuits of size $s(2^{\Omega(n)})$ over F.

Theorem (Second Theorem)

Suppose every circuit family computing $E_{\Pi,n}(x)$ over F, char(F) > 2, has size $s(2^{\Omega(s(n^{O(1)}))})$ for some $s(m) \geq (\log m)^2$. Then permanent polynomial family requires arithmetic circuits of size $s(2^{\Omega(n)})$ over $\mathbb{Z}$.

A weaker version of second theorem was recently shown by Pascal Koiran.

Outline

Multilinear Version of $E_{\Sigma,n}(x)$

- Let $E_{\Sigma,n}(x) = \sum_{t=0}^{(3n^2+n)/2} c_t x^t$.
- Define

$$M_n(z_1, z_2, \ldots, z_u) = \sum_{t=0}^{(3n^2+n)/2} c_t \prod_{j=1}^{u} z_j^{t[j]},$$

where $u = \lceil \log(3n^2 + n) \rceil - 1$, $t[j]$ is jth bit of t, and $c_t \in \{-1, 0, 1\}$ such that

$$E_{\Sigma,n}(x) = M_n(x, x^2, x^{2^2}, \ldots, x^{2^{u-1}}).$$

- The coefficient c_t is computable in polynomial time given t: check if $t = \frac{1}{2}(3m^2 \pm m)$ for some m; if it is, then $c_t = \pm 1$, else $c_t = 0$.
- Using Valiant's result on hardness of permanent, we get that $2^{c \log n} M_n(z_1, z_2, \ldots, z_u)$ can be expressed as permanent of a matrix of size $O(\log n)$ for a suitable choice of constant $c > 0$.

Multilinear Version of $E_{\Sigma,n}(x)$

- Let $E_{\Sigma,n}(x) = \sum_{t=0}^{(3n^2+n)/2} c_t x^t$.
- Define

$$M_n(z_1, z_2, \ldots, z_u) = \sum_{t=0}^{(3n^2+n)/2} c_t \prod_{j=1}^{u} z_j^{t[j]},$$

where $u = \lceil \log(3n^2 + n) \rceil - 1$, $t[j]$ is jth bit of t, and $c_t \in \{-1, 0, 1\}$ such that

$$E_{\Sigma,n}(x) = M_n(x, x^2, x^{2^2}, \ldots, x^{2^{u-1}}).$$

- The coefficient c_t is computable in polynomial time given t: check if $t = \frac{1}{2}(3m^2 \pm m)$ for some m; if it is, then $c_t = \pm 1$, else $c_t = 0$.
- Using Valiant's result on hardness of permanent, we get that $2^{c \log n} M_n(z_1, z_2, \ldots, z_u)$ can be expressed as permanent of a matrix of size $O(\log n)$ for a suitable choice of constant $c > 0$.

Multilinear Version of $E_{\Sigma,n}(x)$

- Let $E_{\Sigma,n}(x) = \sum_{t=0}^{(3n^2+n)/2} c_t x^t$.
- Define

$$M_n(z_1, z_2, \ldots, z_u) = \sum_{t=0}^{(3n^2+n)/2} c_t \prod_{j=1}^{u} z_j^{t[j]},$$

where $u = \lceil \log(3n^2+n) \rceil - 1$, $t[j]$ is jth bit of t, and $c_t \in \{-1, 0, 1\}$ such that

$$E_{\Sigma,n}(x) = M_n(x, x^2, x^{2^2}, \ldots, x^{2^{u-1}}).$$

- The coefficient c_t is computable in polynomial time given t: check if $t = \frac{1}{2}(3m^2 \pm m)$ for some m; if it is, then $c_t = \pm 1$, else $c_t = 0$.
- Using Valiant's result on hardness of permanent, we get that $2^{c \log n} M_n(z_1, z_2, \ldots, z_u)$ can be expressed as permanent of a matrix of size $O(\log n)$ for a suitable choice of constant $c > 0$.

Computing $E_{\Sigma,n}(x)$

- Suppose permanent family can be computed by a circuit family of size $s(2^{o(n)})$ over F.
- Then, the polynomial family $2^{c \log n} M_n$ can be computed by a circuit family of size $s(n^{o(1)})$.
- Let circuit C compute $2^{c \log n} M_n$.
- Modify C by replacing its input z_j by x^{2^j}.
- This adds $O(\log n)$ multiplication gates to C.
- Multiply the resulting circuit by $2^{-c \log n}$ in F (since $\mathrm{char}(F) > 2$, it always exists).
- The final circuit computes $E_{\Sigma,n}(x)$ and has size $s(n^{o(1)})$, a contradiction.

Computing $E_{\Sigma,n}(x)$

- Suppose permanent family can be computed by a circuit family of size $s(2^{o(n)})$ over F.
- Then, the polynomial family $2^{c \log n} M_n$ can be computed by a circuit family of size $s(n^{o(1)})$.
- Let circuit C compute $2^{c \log n} M_n$.
- Modify C by replacing its input z_j by x^{2^j}.
- This adds $O(\log n)$ multiplication gates to C.
- Multiply the resulting circuit by $2^{-c \log n}$ in F (since $\mathrm{char}(F) > 2$, it always exists).
- The final circuit computes $E_{\Sigma,n}(x)$ and has size $s(n^{o(1)})$, a contradiction.

Computing $E_{\Sigma,n}(x)$

- Suppose permanent family can be computed by a circuit family of size $s(2^{o(n)})$ over F.
- Then, the polynomial family $2^{c \log n} M_n$ can be computed by a circuit family of size $s(n^{o(1)})$.
- Let circuit C compute $2^{c \log n} M_n$.
- Modify C by replacing its input z_j by x^{2^j}.
- This adds $O(\log n)$ multiplication gates to C.
- Multiply the resulting circuit by $2^{-c \log n}$ in F (since char$(F) > 2$, it always exists).
- The final circuit computes $E_{\Sigma,n}(x)$ and has size $s(n^{o(1)})$, a contradiction.

COMPUTING $E_{\Sigma,n}(x)$

- Suppose permanent family can be computed by a circuit family of size $s(2^{o(n)})$ over F.
- Then, the polynomial family $2^{c \log n} M_n$ can be computed by a circuit family of size $s(n^{o(1)})$.
- Let circuit C compute $2^{c \log n} M_n$.
- Modify C by replacing its input z_j by x^{2^j}.
- This adds $O(\log n)$ multiplication gates to C.
- Multiply the resulting circuit by $2^{-c \log n}$ in F (since $\mathrm{char}(F) > 2$, it always exists).
- The final circuit computes $E_{\Sigma,n}(x)$ and has size $s(n^{o(1)})$, a contradiction.

Outline

Setup

- Assume that there is a circuit family of size $s(2^{o(n)})$ computing permanent polynomial over $\mathbb{Z}$.
- Let $P(x) = E_{\Pi,n}(x)$ for some $n > 1$.
- Degree of $P(x)$ equals $\frac{1}{2}n(n+1) < n^2$.
- Let char$(F) = p$. Since coefficients of $P(x)$ are in F_p, we can assume $F = F_p$.
- Let $\hat{F}$ be an extension of F with $n^2 \leq q = |\hat{F}| = O(n^2)$.

Setup

- Assume that there is a circuit family of size $s(2^{o(n)})$ computing permanent polynomial over $\mathbb{Z}$.
- Let $P(x) = E_{\Pi,n}(x)$ for some $n > 1$.
- Degree of $P(x)$ equals $\frac{1}{2}n(n+1) < n^2$.
- Let $\mathrm{char}(F) = p$. Since coefficients of $P(x)$ are in F_p, we can assume $F = F_p$.
- Let $\hat{F}$ be an extension of F with $n^2 \leq q = |\hat{F}| = O(n^2)$.

Setup

- Assume that there is a circuit family of size $s(2^{o(n)})$ computing permanent polynomial over $\mathbb{Z}$.
- Let $P(x) = E_{\Pi,n}(x)$ for some $n > 1$.
- Degree of $P(x)$ equals $\frac{1}{2}n(n+1) < n^2$.
- Let $\mathrm{char}(F) = p$. Since coefficients of $P(x)$ are in F_p, we can assume $F = F_p$.
- Let $\hat{F}$ be an extension of F with $n^2 \leq q = |\hat{F}| = O(n^2)$.

An Alternative Expression for $P(x)$

- By Langrange's interpolation formula, we have:

$$P(x) = \sum_{\alpha \in \hat{F}} P(\alpha) \cdot \frac{\prod_{\beta \in \hat{F}, \beta \neq \alpha} (x - \beta)}{\prod_{\beta \in \hat{F}, \beta \neq \alpha} (\alpha - \beta)}.$$

- Observe that

$$\prod_{\beta \in \hat{F}, \beta \neq \alpha} (\alpha - \beta) = \prod_{\beta \in \hat{F}^*} \beta = -1,$$

and

$$\begin{aligned}\prod_{\beta \in \hat{F}, \beta \neq \alpha} (x - \beta) &= \frac{\prod_{\beta \in \hat{F}} (x - \beta)}{x - \alpha} \\ &= \frac{x^q - x}{x - \alpha} = \sum_{j=1}^{q-1} \alpha^{j-1} x^{q-j}.\end{aligned}$$

An Alternative Expression for $P(x)$

- By Langrange's interpolation formula, we have:

$$P(x) = \sum_{\alpha \in \hat{F}} P(\alpha) \cdot \frac{\prod_{\beta \in \hat{F}, \beta \neq \alpha} (x - \beta)}{\prod_{\beta \in \hat{F}, \beta \neq \alpha} (\alpha - \beta)}.$$

- Observe that

$$\prod_{\beta \in \hat{F}, \beta \neq \alpha} (\alpha - \beta) = \prod_{\beta \in \hat{F}^*} \beta = -1,$$

and

$$\begin{aligned} \prod_{\beta \in \hat{F}, \beta \neq \alpha} (x - \beta) &= \frac{\prod_{\beta \in \hat{F}} (x - \beta)}{x - \alpha} \\ &= \frac{x^q - x}{x - \alpha} = \sum_{j=1}^{q-1} \alpha^{j-1} x^{q-j}. \end{aligned}$$

An Alternative Expression for $P(x)$

- Therefore,

$$\begin{aligned} P(x) &= -\sum_{\alpha \in \hat{F}} P(\alpha) \sum_{j=1}^{q-1} \alpha^{j-1} x^{q-j} \\ &= -\sum_{j=1}^{q-1} \Big(\sum_{\alpha \in \hat{F}} P(\alpha) \alpha^{j-1} \Big) x^{q-j}. \end{aligned}$$

- Now if we can compute $P(\alpha)$ efficiently, we can compute $P(x)$ as permanent of a small size matrix.
- However, as

$$P(\alpha) = \prod_{m=1}^{n} (1 - \alpha^m),$$

we cannot compute it directly.

An Alternative Expression for $P(x)$

- Therefore,

$$\begin{aligned} P(x) &= -\sum_{\alpha \in \hat{F}} P(\alpha) \sum_{j=1}^{q-1} \alpha^{j-1} x^{q-j} \\ &= -\sum_{j=1}^{q-1} \Big(\sum_{\alpha \in \hat{F}} P(\alpha)\alpha^{j-1}\Big) x^{q-j}. \end{aligned}$$

- Now if we can compute $P(\alpha)$ efficiently, we can compute $P(x)$ as permanent of a small size matrix.
- However, as

$$P(\alpha) = \prod_{m=1}^{n} (1 - \alpha^m),$$

we cannot compute it directly.

Computing $P(\alpha)$

- Let g be a generator of $\hat{F}^*$.
- Define NTM N as: on input α, guess t and m with $0 \leq t < q$ and $1 \leq m \leq n$. Check if $g^t = 1 - \alpha^m$. If yes, output t on the part, else output 0.
- N is a polynomial time TM, and

$$\#N(\alpha) = \sum_{m=1}^{n} t_m,$$

  where $g^{t_m} = 1 - \alpha^m$.
- Hence, $g^{\#N(\alpha)} = P(\alpha)$.
- Therefore, $P(\alpha)$ is computable in $\mathrm{P}^{\#\mathrm{P}}$.

Computing $P(\alpha)$

- Let g be a generator of $\hat{F}^*$.
- Define NTM N as: on input α, guess t and m with $0 \leq t < q$ and $1 \leq m \leq n$. Check if $g^t = 1 - \alpha^m$. If yes, output t on the part, else output 0.
- N is a polynomial time TM, and

$$\#N(\alpha) = \sum_{m=1}^{n} t_m,$$

  where $g^{t_m} = 1 - \alpha^m$.
- Hence, $g^{\#N(\alpha)} = P(\alpha)$.
- Therefore, $P(\alpha)$ is computable in $\mathrm{P}^{\#\mathrm{P}}$.

Computing $P(\alpha)$

- Let g be a generator of $\hat{F}^*$.
- Define NTM N as: on input α, guess t and m with $0 \leq t < q$ and $1 \leq m \leq n$. Check if $g^t = 1 - \alpha^m$. If yes, output t on the part, else output 0.
- N is a polynomial time TM, and

$$\#N(\alpha) = \sum_{m=1}^{n} t_m,$$

  where $g^{t_m} = 1 - \alpha^m$.
- Hence, $g^{\#N(\alpha)} = P(\alpha)$.
- Therefore, $P(\alpha)$ is computable in $\mathrm{P}^{\#\mathrm{P}}$.

Computing $P(x)$

- Since permanent is complete for $\#\mathrm{P}$, we get that $P(\alpha)$ can be computed by *boolean* circuits of size $s(n^{o(1)})$.
- Therefore, $P(x)$ can be computed by *arithmetic* circuits of size $s(2^{o(s(n^{o(1)}))})$ over F.
- A Contradiction.

Computing $P(x)$

- Since permanent is complete for $\#\mathrm{P}$, we get that $P(\alpha)$ can be computed by *boolean* circuits of size $s(n^{o(1)})$.
- Therefore, $P(x)$ can be computed by *arithmetic* circuits of size $s(2^{o(s(n^{o(1)}))})$ over F.
- A Contradiction.

Outline

Polynomial Identity Testing Problem

PIT over F

Given an arithmetic circuit over field F, determine if the polynomial computed by the circuit is identically zero.

- Admits a number of randomized polynomial time algorithms but no deterministic one is known.
- Has an interesting connection with hardness of computing $E_{\Pi,n}(x)$.

Polynomial Identity Testing Problem

PIT over F

Given an arithmetic circuit over field F, determine if the polynomial computed by the circuit is identically zero.

- Admits a number of randomized polynomial time algorithms but no deterministic one is known.
- Has an interesting connection with hardness of computing $E_{\Pi,n}(x)$.

Computing Multiples of $E_{\Pi,n}(x)$

Let $P_m(x)$ be a family of polynomials with $P_m(x)$ of degree $m^{O(1)}$. The family is an $n(m)$-*multiple* of the family $E_{\Pi,n}(x)$ if for every m, $E_{\Pi,n(m)}(x)$ divides $P_m(x)$.

- It is possible that $E_{\Pi,n}(x)$ requires circuit of size $n^{\Omega(1)}$ to compute.
- Does it also mean that every $n(m)$-multiple of $E_{\Pi,n}(x)$ also requires circuits of size $(n(m))^{\Omega(1)}$ to compute?
- If yes, we get a black-box derandomization of PIT.

Computing Multiples of $E_{\Pi,n}(x)$

Let $P_m(x)$ be a family of polynomials with $P_m(x)$ of degree $m^{O(1)}$. The family is an $n(m)$-*multiple* of the family $E_{\Pi,n}(x)$ if for every m, $E_{\Pi,n(m)}(x)$ divides $P_m(x)$.

- It is possible that $E_{\Pi,n}(x)$ requires circuit of size $n^{\Omega(1)}$ to compute.
- Does it also mean that every $n(m)$-multiple of $E_{\Pi,n}(x)$ also requires circuits of size $(n(m))^{\Omega(1)}$ to compute?
- If yes, we get a black-box derandomization of PIT.

Derandomization of PIT

Theorem

If every $n(m)$-multiple of $E_{\Pi,n}(x)$, for every $n(m) = m^{O(1)}$, requires circuits of size $(n(m))^{\Omega(1)}$ to compute over field F, then there exists a polynomial-time black-box derandomization of PIT over F.

Proof

- Assume that every $n(m)$-multiple of $E_{\Pi,n}(x)$ requires circuits of size $(n(m))^{\delta}$ for some $\delta > 0$.
- Let C be an arithmetic circuit of size m computing a polynomial $Q(y_1, \ldots, y_m)$ over F.
- The degree of Q is bounded by 2^m.
- We give a polynomial time algorithm for checking if Q is identically zero.

Proof

- Assume that every $n(m)$-multiple of $E_{\Pi,n}(x)$ requires circuits of size $(n(m))^\delta$ for some $\delta > 0$.
- Let C be an arithmetic circuit of size m computing a polynomial $Q(y_1, \ldots, y_m)$ over F.
- The degree of Q is bounded by 2^m.
- We give a polynomial time algorithm for checking if Q is identically zero.

Proof

- Assume that every $n(m)$-multiple of $E_{\Pi,n}(x)$ requires circuits of size $(n(m))^\delta$ for some $\delta > 0$.
- Let C be an arithmetic circuit of size m computing a polynomial $Q(y_1, \dots, y_m)$ over F.
- The degree of Q is bounded by 2^m.
- We give a polynomial time algorithm for checking if Q is identically zero.

The Algorithm

- Let $D = 2^m + 1$ and replace y^i by $x^{D^{i-1}}$ as input to C.
- This requires an additional $O(m^2)$ wires at the bottom of C.
- Let the resulting circuit be $\hat{C}$, and $R(x)$ be the polynomial computed by it.
- The size of $\hat{C}$ is $O(m^2)$ and the degree of $R(x)$ is at most 2^{m^2}.
- It is easy to see that $R(x)$ is non-zero iff $Q(y_1, \ldots, y_m)$ is.
- Test if $R(x) = 0 \ (mod\ (x^\ell - 1)^k)$ for $1 \leq \ell \leq n = m^{3/\delta}$ and k is the largest number such that $(x^\ell - 1)^k$ divides $E_{\Pi,n}(x)$.
- Output ZERO iff all the tests succeed.

The Algorithm

- Let $D = 2^m + 1$ and replace y^i by $x^{D^{i-1}}$ as input to C.
- This requires an additional $O(m^2)$ wires at the bottom of C.
- Let the resulting circuit be $\hat{C}$, and $R(x)$ be the polynomial computed by it.
- The size of $\hat{C}$ is $O(m^2)$ and the degree of $R(x)$ is at most 2^{m^2}.
- It is easy to see that $R(x)$ is non-zero iff $Q(y_1, \ldots, y_m)$ is.
- Test if $R(x) = 0 \ (mod\ (x^\ell - 1)^k)$ for $1 \leq \ell \leq n = m^{3/\delta}$ and k is the largest number such that $(x^\ell - 1)^k$ divides $E_{\Pi,n}(x)$.
- Output ZERO iff all the tests succeed.

The Algorithm

- Let $D = 2^m + 1$ and replace y^i by $x^{D^{i-1}}$ as input to C.
- This requires an additional $O(m^2)$ wires at the bottom of C.
- Let the resulting circuit be $\hat{C}$, and $R(x)$ be the polynomial computed by it.
- The size of $\hat{C}$ is $O(m^2)$ and the degree of $R(x)$ is at most 2^{m^2}.
- It is easy to see that $R(x)$ is non-zero iff $Q(y_1, \ldots, y_m)$ is.
- Test if $R(x) = 0 \ (mod\ (x^\ell - 1)^k)$ for $1 \leq \ell \leq n = m^{3/\delta}$ and k is the largest number such that $(x^\ell - 1)^k$ divides $E_{\Pi,n}(x)$.
- Output ZERO iff all the tests succeed.

The Algorithm

- Let $D = 2^m + 1$ and replace y^i by $x^{D^{i-1}}$ as input to C.
- This requires an additional $O(m^2)$ wires at the bottom of C.
- Let the resulting circuit be $\hat{C}$, and $R(x)$ be the polynomial computed by it.
- The size of $\hat{C}$ is $O(m^2)$ and the degree of $R(x)$ is at most 2^{m^2}.
- It is easy to see that $R(x)$ is non-zero iff $Q(y_1, \ldots, y_m)$ is.
- Test if $R(x) = 0 \ (mod\ (x^\ell - 1)^k)$ for $1 \leq \ell \leq n = m^{3/\delta}$ and k is the largest number such that $(x^\ell - 1)^k$ divides $E_{\Pi,n}(x)$.
- Output ZERO iff all the tests succeed.

Correctness

- The algorithm is clearly deterministic, polynomial-time, and black-box.
- Observe that if all the tests succeed, it implies that $E_{\Pi,n}(x)$ divides $R(x)$.
- If $R(x)$ is non-zero then, by our assumption on n-multiples of $E_{\Pi,n}(x)$, $R(x)$ requires a circuit of size $n^{\delta} = m^3$ to compute.
- However, circuit $\hat{C}$, of size $O(m^2)$, computes $R(x)$.
- A contradiction.

Correctness

- The algorithm is clearly deterministic, polynomial-time, and black-box.
- Observe that if all the tests succeed, it implies that $E_{\Pi,n}(x)$ divides $R(x)$.
- If $R(x)$ is non-zero then, by our assumption on n-multiples of $E_{\Pi,n}(x)$, $R(x)$ requires a circuit of size $n^\delta = m^3$ to compute.
- However, circuit $\hat{C}$, of size $O(m^2)$, computes $R(x)$.
- A contradiction.

Correctness

- The algorithm is clearly deterministic, polynomial-time, and black-box.
- Observe that if all the tests succeed, it implies that $E_{\Pi,n}(x)$ divides $R(x)$.
- If $R(x)$ is non-zero then, by our assumption on n-multiples of $E_{\Pi,n}(x)$, $R(x)$ requires a circuit of size $n^\delta = m^3$ to compute.
- However, circuit $\hat{C}$, of size $O(m^2)$, computes $R(x)$.
- A contradiction.

Correctness

- The algorithm is clearly deterministic, polynomial-time, and black-box.
- Observe that if all the tests succeed, it implies that $E_{\Pi,n}(x)$ divides $R(x)$.
- If $R(x)$ is non-zero then, by our assumption on n-multiples of $E_{\Pi,n}(x)$, $R(x)$ requires a circuit of size $n^{\delta} = m^3$ to compute.
- However, circuit $\hat{C}$, of size $O(m^2)$, computes $R(x)$.
- A contradiction.

Outline

Open Questions

Several questions remain open:

1. Is the polynomial $E_{\Pi,n}(x)$ computable over F_p in $\mathrm{Mod}_p\mathrm{P}$?
2. Does $E_{\Pi,n}(x)$ require circuits of size $n^{\Omega(1)}$?
3. Does every n-multiple of $E_{\Pi,n}(x)$ requires circuits of size $n^{\Omega(1)}$?

Open Questions

Several questions remain open:

1. Is the polynomial $E_{\Pi,n}(x)$ computable over F_p in Mod_pP?
2. Does $E_{\Pi,n}(x)$ require circuits of size $n^{\Omega(1)}$?
3. Does every n-multiple of $E_{\Pi,n}(x)$ requires circuits of size $n^{\Omega(1)}$?

Open Questions

Several questions remain open:

1. Is the polynomial $E_{\Pi,n}(x)$ computable over F_p in $\mathrm{Mod}_p\mathrm{P}$?
2. Does $E_{\Pi,n}(x)$ require circuits of size $n^{\Omega(1)}$?
3. Does every n-multiple of $E_{\Pi,n}(x)$ requires circuits of size $n^{\Omega(1)}$?

A Conjecture

Conjecture

Answer to all three questions above is yes.

If the conjecture is true then an exponential lower bound on permanent polynomial follows.

A Conjecture

Conjecture

Answer to all three questions above is yes.

If the conjecture is true then an exponential lower bound on permanent polynomial follows.

Thoughts on the Conjecture

- The conjecture relates the size of an arithmetic circuit computing a polynomial to the number of distinct small roots of unity that the polynomial can have.
- It is similar in spirit to τ-conjecture of Shub-Smale that relates the size of an arithmetic circuit computing a polynomial to the number of integer roots the polynomial can have.

Thoughts on the Conjecture

- The conjecture relates the size of an arithmetic circuit computing a polynomial to the number of distinct small roots of unity that the polynomial can have.
- It is similar in spirit to τ-conjecture of Shub-Smale that relates the size of an arithmetic circuit computing a polynomial to the number of integer roots the polynomial can have.