On the Arithmetic Complexity of Euler Function

Manindra Agrawal

IIT Kanpur

Bangalore, Sep 2010

Euler Function

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

Defined by Leonhard Euler.

Irrelevant Facts

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

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

Irrelevant Facts

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

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

Irrelevant Facts

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.

Irrelevant Facts

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.

Irrelevant Facts

Over Complex Plane

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

- Undefined outside unit disk.
- Zero at unit circle.
- Bounded inside the unit disk.

Proof. Straightforward.

Irrelevant Facts

Over Complex Plane

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

- Undefined outside unit disk.
- Zero at unit circle.
- Bounded inside the unit disk.

Proof. Straightforward.

Irrelevant Facts

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.

Irrelevant Facts

Dedekind Eta Function

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

$\eta(z)$ 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.

Computing Euler Function

- Let
$$E_n(x) = \prod_{k=1}^{n} (1 - x^k).$$
- So, $E(x) = \lim_{n \mapsto \infty} E_n(x)$.
- A circuit family computing $E_n(x)$ can be viewed as computing $E(x)$.
- We will consider arithmetic circuits for computing $E_n(x)$.

Computing Euler Function

- Let

$$E_n(x) = \prod_{k=1}^{n} (1 - x^k).$$

- So, $E(x) = \lim_{n \mapsto \infty} E_n(x)$.
- A circuit family computing $E_n(x)$ can be viewed as computing $E(x)$.
- We will consider arithmetic circuits for computing $E_n(x)$.

Circuits for $E_n(x)$

- A circuit computing $E_n(x)$ over field F takes as input x and -1; and outputs $E_n(x)$.
- It is allowed to use addition and multiplication gates of arbitrary fanin over F.
- Size of a circuit is the number of gates in it (not the number of wires).
- A depth three circuit of size $\Theta(n)$ can compute $E_n(x)$ over any field F: follows from definition.

Circuits for $E_n(x)$

- A circuit computing $E_n(x)$ over field F takes as input x and -1; and outputs $E_n(x)$.
- It is allowed to use addition and multiplication gates of arbitrary fanin over F.
- Size of a circuit is the number of gates in it (not the number of wires).
- A depth three circuit of size $\Theta(n)$ can compute $E_n(x)$ over any field F: follows from definition.

Circuits for $E_n(x)$

- Can a depth three or four circuit do significantly better?
- It is not clear.
- A proof of this for a field of finite characteristic gives a superpolynomial lower bound on computing permanent polynomial family by arithmetic circuits.

Circuits for $E_n(x)$

- Can a depth three or four circuit do significantly better?
- It is not clear.
- A proof of this for a field of finite characteristic gives a superpolynomial lower bound on computing permanent polynomial family by arithmetic circuits.

The Main Theorem

Theorem

Suppose every depth four circuit family computing $E_n(x)$ over F, char(F) > 0, has size at least n^ϵ, for some fixed $\epsilon > 0$. Then permanent polynomial family cannot be computed by polynomial-size arithmetic circuits over $\mathbb{Z}$.

Similar results have been obtained recently by Pascal Koiran.

The Main Theorem

Theorem

Suppose every depth four circuit family computing $E_n(x)$ over F, char(F) > 0, has size at least n^ϵ, for some fixed $\epsilon > 0$. Then permanent polynomial family cannot be computed by polynomial-size arithmetic circuits over $\mathbb{Z}$.

Similar results have been obtained recently by Pascal Koiran.

Proof

- Without loss of generality, we can assume that the depth four circuit family computes $E_n(x)$ over F with $F = F_p$ for some prime p.
  - Follows from the fact that circuits over an extension field of F_p can be simulated by circuits over F_p with only a small increase in size.
- Assume that there is a polynomial-size circuit family computing permanent polynomials over $\mathbb{Z}$.

Proof

- Without loss of generality, we can assume that the depth four circuit family computes $E_n(x)$ over F with $F = F_p$ for some prime p.
  - Follows from the fact that circuits over an extension field of F_p can be simulated by circuits over F_p with only a small increase in size.
- Assume that there is a polynomial-size circuit family computing permanent polynomials over $\mathbb{Z}$.

Proof: An Alternative Expression for $E_n(x)$

- Let $\hat{F}$ be an extension of F with $t = |\hat{F}| \geq n^2$ and $t = O(n^2)$.
- Let $c_\alpha = E_n(\alpha)$ for every $\alpha \in \hat{F}$.
- Define $G(x)$ as:

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

- $G_n(x)$ agrees with $E_n(x)$ at every point in $\hat{F}$.
- And $G_n(x) - E_n(x)$ is a polynomial of degree $< t$.
- Therefore, $E_n(x) = G_n(x)$.

Proof: An Alternative Expression for $E_n(x)$

- Let $\hat{F}$ be an extension of F with $t = |\hat{F}| \geq n^2$ and $t = O(n^2)$.
- Let $c_\alpha = E_n(\alpha)$ for every $\alpha \in \hat{F}$.
- Define $G(x)$ as:

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

- $G_n(x)$ agrees with $E_n(x)$ at every point in $\hat{F}$.
- And $G_n(x) - E_n(x)$ is a polynomial of degree $< t$.
- Therefore, $E_n(x) = G_n(x)$.

Proof: An Alternative Expression for $E_n(x)$

- Let $\hat{F}$ be an extension of F with $t = |\hat{F}| \geq n^2$ and $t = O(n^2)$.
- Let $c_\alpha = E_n(\alpha)$ for every $\alpha \in \hat{F}$.
- Define $G(x)$ as:

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

- $G_n(x)$ agrees with $E_n(x)$ at every point in $\hat{F}$.
- And $G_n(x) - E_n(x)$ is a polynomial of degree $< t$.
- Therefore, $E_n(x) = G_n(x)$.

Proof: An Alternative Expression for $E_n(x)$

- Let $\hat{F}$ be an extension of F with $t = |\hat{F}| \geq n^2$ and $t = O(n^2)$.
- Let $c_\alpha = E_n(\alpha)$ for every $\alpha \in \hat{F}$.
- Define $G(x)$ as:

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

- $G_n(x)$ agrees with $E_n(x)$ at every point in $\hat{F}$.
- And $G_n(x) - E_n(x)$ is a polynomial of degree $< t$.
- Therefore, $E_n(x) = G_n(x)$.

Proof: An Alternative Expression for $E_n(x)$

- Let $\hat{F}$ be an extension of F with $t = |\hat{F}| \geq n^2$ and $t = O(n^2)$.
- Let $c_\alpha = E_n(\alpha)$ for every $\alpha \in \hat{F}$.
- Define $G(x)$ as:

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

- $G_n(x)$ agrees with $E_n(x)$ at every point in $\hat{F}$.
- And $G_n(x) - E_n(x)$ is a polynomial of degree $< t$.
- Therefore, $E_n(x) = G_n(x)$.

Proof: Computing $G_n(x)$

- Let g be a generator of $\hat{F}^*$.
- Rewrite $G_n(x)$ as:

$$G_n(x) = x - x^{t-1} + \sum_{k=0}^{t-2} c_{g^k} \frac{\prod_{\beta \in \hat{F}, \beta \neq g^k}(x - \beta)}{\prod_{\beta \in \hat{F}, \beta \neq g^k}(g^k - \beta)}$$

$$= \sum_{k=0}^{t-1} u(n,k) x^k.$$

- We show that the function u belongs to $\#P^{\#P}$.
- The size of inputs in computations below is $O(\log n)$.
- Notice that

$$c_{g^k} = \prod_{\ell=1}^{n}(1 - g^{k\ell}) = g^{\sum_{\ell=1}^{n} h_\ell},$$

for appropriate numbers h_ℓ.
- From ℓ and k, numbers h_ℓ can be computed by a single-valued NP machine.

Proof: Computing $G_n(x)$

- Let g be a generator of $\hat{F}^*$.
- Rewrite $G_n(x)$ as:

$$\begin{aligned} G_n(x) &= x - x^{t-1} + \sum_{k=0}^{t-2} c_{g^k} \frac{\prod_{\beta \in \hat{F}, \beta \neq g^k} (x - \beta)}{\prod_{\beta \in \hat{F}, \beta \neq g^k} (g^k - \beta)} \\ &= \sum_{k=0}^{t-1} u(n,k) x^k. \end{aligned}$$

- We show that the function u belongs to $\#\mathrm{P}^{\#\mathrm{P}}$.
- The size of inputs in computations below is $O(\log n)$.
- Notice that

$$c_{g^k} = \prod_{\ell=1}^{n} (1 - g^{k\ell}) = g^{\sum_{\ell=1}^{n} h_\ell},$$

for appropriate numbers h_ℓ.
- From ℓ and k, numbers h_ℓ can be computed by a single-valued NP machine.

Proof: Computing $G_n(x)$

- Let g be a generator of $\hat{F}^*$.
- Rewrite $G_n(x)$ as:

$$G_n(x) = x - x^{t-1} + \sum_{k=0}^{t-2} c_{g^k} \frac{\prod_{\beta \in \hat{F}, \beta \neq g^k} (x - \beta)}{\prod_{\beta \in \hat{F}, \beta \neq g^k} (g^k - \beta)}$$
$$= \sum_{k=0}^{t-1} u(n,k) x^k.$$

- We show that the function u belongs to $\#\mathrm{P}^{\#\mathrm{P}}$.
- The size of inputs in computations below is $O(\log n)$.
- Notice that

$$c_{g^k} = \prod_{\ell=1}^{n} (1 - g^{k\ell}) = g^{\sum_{\ell=1}^{n} h_\ell},$$

for appropriate numbers h_ℓ.

- From ℓ and k, numbers h_ℓ can be computed by a single-valued NP machine.

Proof: Computing $G_n(x)$

- Let g be a generator of $\hat{F}^*$.
- Rewrite $G_n(x)$ as:

$$
\begin{aligned}
G_n(x) &= x - x^{t-1} + \sum_{k=0}^{t-2} c_{g^k} \frac{\prod_{\beta \in \hat{F}, \beta \neq g^k}(x-\beta)}{\prod_{\beta \in \hat{F}, \beta \neq g^k}(g^k-\beta)} \\
&= \sum_{k=0}^{t-1} u(n,k) x^k.
\end{aligned}
$$

- We show that the function u belongs to $\#\mathrm{P}^{\#\mathrm{P}}$.
- The size of inputs in computations below is $O(\log n)$.
- Notice that

$$
c_{g^k} = \prod_{\ell=1}^{n}(1 - g^{k\ell}) = g^{\sum_{\ell=1}^{n} h_\ell},
$$

  for appropriate numbers h_ℓ.
- From ℓ and k, numbers h_ℓ can be computed by a single-valued NP machine.

Proof: Computing $G_n(x)$

- Observe that

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

- And

$$\begin{aligned}
\prod_{\beta \in \hat{F}, \beta \neq g^k} (x - \beta) &= \frac{\prod_{\beta \in \hat{F}} (x - \beta)}{x - g^k} \\
&= \frac{x^t - x}{x - g^k} \\
&= x^{t-1} + g^k x^{t-2} + g^{2k} x^{t-3} + \cdots + g^{(t-2)k} x,
\end{aligned}$$

for $0 \leq k < t$.

Proof: Computing $G_n(x)$

- Observe that

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

- And

$$\begin{aligned}
\prod_{\beta \in \hat{F}, \beta \neq g^k} (x - \beta) &= \frac{\prod_{\beta \in \hat{F}} (x - \beta)}{x - g^k} \\
&= \frac{x^t - x}{x - g^k} \\
&= x^{t-1} + g^k x^{t-2} + g^{2k} x^{t-3} + \cdots + g^{(t-2)k} x,
\end{aligned}$$

for $0 \leq k < t$.

Proof: Computing $G_n(x)$

- Hence,

$$
\begin{aligned}
G_n(x) &= x - x^{t-1} - \sum_{k=0}^{t-2} c_{g^k} \cdot \sum_{\ell=1}^{t-1} g^{(t-\ell-1)k} x^\ell \\
&= x - x^{t-1} - \sum_{\ell=1}^{t-1} (\sum_{k=0}^{t-2} c_{g^k} g^{(t-\ell-1)k}) x^\ell \\
&= x - x^{t-1} - \sum_{\ell=1}^{t-1} (\sum_{k=0}^{t-2} g^{(t-\ell-1)k + \sum_{m=1}^{n} h_m}) x^\ell .
\end{aligned}
$$

- The above equation shows that the function u is in $\#P^{\#P}$.

Proof: Computing $G_n(x)$

- Hence,

$$\begin{aligned} G_n(x) &= x - x^{t-1} - \sum_{k=0}^{t-2} c_{g^k} \cdot \sum_{\ell=1}^{t-1} g^{(t-\ell-1)k} x^\ell \\ &= x - x^{t-1} - \sum_{\ell=1}^{t-1} (\sum_{k=0}^{t-2} c_{g^k} g^{(t-\ell-1)k}) x^\ell \\ &= x - x^{t-1} - \sum_{\ell=1}^{t-1} (\sum_{k=0}^{t-2} g^{(t-\ell-1)k + \sum_{m=1}^{n} h_m}) x^\ell. \end{aligned}$$

- The above equation shows that the function u is in $\#P^{\#P}$.

Proof: Computing $G_n(x)$

- We have assumed that the permanent polynomial can be computed by a polynomial size circuit.
- This implies that any function in #P can be computed by a polynomial size arithmetic circuit.
- This implies that the function u is in #P/poly.
- Since
$$G_n(x) = \sum_{k=0}^{t} u(n,k)x^k,$$
it follows that $G_n(x)$ can be computed as permanent of a small size $(= O(\log n))$ matrix.
- This matrix will have entries 0, -1, and following powers of x: x, x^2, x^{2^2}, x^{2^3}, ..., $x^{2^{\lceil \log t \rceil}}$:
  - permanent of a matrix is a multilinear polynomial of its entries, and so these powers of x can be used to create all the other powers of $x < t$.
- This gives $\log^{O(1)} n$-size circuit to compute $G_n(x)$ over Z.

Proof: Computing $G_n(x)$

- We have assumed that the permanent polynomial can be computed by a polynomial size circuit.
- This implies that any function in #P can be computed by a polynomial size arithmetic circuit.
- This implies that the function u is in #P/poly.
- Since

$$G_n(x) = \sum_{k=0}^{t} u(n,k)x^k,$$

  it follows that $G_n(x)$ can be computed as permanent of a small size $(= O(\log n))$ matrix.
- This matrix will have entries 0, -1, and following powers of x: x, x^2, x^{2^2}, x^{2^3}, ..., $x^{2^{\lceil \log t \rceil}}$:
  - permanent of a matrix is a multilinear polynomial of its entries, and so these powers of x can be used to create all the other powers of $x < t$.
- This gives $\log^{O(1)} n$-size circuit to compute $G_n(x)$ over Z.

Proof: Computing $G_n(x)$

- We have assumed that the permanent polynomial can be computed by a polynomial size circuit.
- This implies that any function in #P can be computed by a polynomial size arithmetic circuit.
- This implies that the function u is in #P/poly.
- Since

$$G_n(x) = \sum_{k=0}^{t} u(n,k)x^k,$$

  it follows that $G_n(x)$ can be computed as permanent of a small size ($= O(\log n)$) matrix.
- This matrix will have entries 0, -1, and following powers of x: x, x^2, x^{2^2}, x^{2^3}, …, $x^{2^{\lceil \log t \rceil}}$:
  - permanent of a matrix is a multilinear polynomial of its entries, and so these powers of x can be used to create all the other powers of $x < t$.
- This gives $\log^{O(1)} n$-size circuit to compute $G_n(x)$ over Z.

Proof: Computing $G_n(x)$

- We have assumed that the permanent polynomial can be computed by a polynomial size circuit.
- This implies that any function in #P can be computed by a polynomial size arithmetic circuit.
- This implies that the function u is in #P/poly.
- Since
$$G_n(x) = \sum_{k=0}^{t} u(n,k)x^k,$$
it follows that $G_n(x)$ can be computed as permanent of a small size $(= O(\log n))$ matrix.
- This matrix will have entries 0, -1, and following powers of x: x, x^2, x^{2^2}, x^{2^3}, $\ldots$, $x^{2^{\lceil \log t \rceil}}$:
  - permanent of a matrix is a multilinear polynomial of its entries, and so these powers of x can be used to create all the other powers of $x < t$.
- This gives $\log^{O(1)} n$-size circuit to compute $G_n(x)$ over Z.

Proof: Computing $E_n(x)$

- This circuit can be converted to a $\log^{O(1)} n$-size arithmetic circuit over F since coefficients of $G_n(x)$ are in F.
- Using [AV08], this circuit can be transformed to a depth four circuit of size $n^{o(1)}$.
- This implies that the polynomial $E_n(x)$ can be computed by a $n^{o(1)}$-size arithmetic circuit over F.
- This contradicts the hypothesis that $E_n(x)$ requires circuit of size n^{ϵ} over F for some $\epsilon > 0$.

Proof: Computing $E_n(x)$

- This circuit can be converted to a $\log^{O(1)} n$-size arithmetic circuit over F since coefficients of $G_n(x)$ are in F.
- Using [AV08], this circuit can be transformed to a depth four circuit of size $n^{o(1)}$.
- This implies that the polynomial $E_n(x)$ can be computed by a $n^{o(1)}$-size arithmetic circuit over F.
- This contradicts the hypothesis that $E_n(x)$ requires circuit of size n^{ϵ} over F for some $\epsilon > 0$.

Generalizations

- The theorem can be strengthened to show that permanent polynomial requires size $s(n)$ where s is any function satisfying $s(s(n)) = 2^{o(n)}$.
- It should be possible to strengthen it further to $s(n) = 2^{\Omega(n)}$.

Generalizations

- The theorem can be strengthened to show that permanent polynomial requires size $s(n)$ where s is any function satisfying $s(s(n)) = 2^{o(n)}$.
- It should be possible to strengthen it further to $s(n) = 2^{\Omega(n)}$.

A Conjecture

Conjecture

Let $P(x)$ be a polynomial computed by a depth four circuit of size m. Then $P(x) \neq 0 \ (mod\ x^k - 1)$ for some $k \leq m^{1/4}$.

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

A Conjecture

Conjecture

Let $P(x)$ be a polynomial computed by a depth four circuit of size m. Then $P(x) \neq 0 \ (mod\ x^k - 1)$ for some $k \leq m^{1/4}$.

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

Thoughts on the Conjecture

- The conjecture relates the size of a shallow circuit computing a polynomial to the number of 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 a shallow circuit computing a polynomial to the number of 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.

Open Problems

- Prove the theorem for $s(n) = 2^{\Omega(n)}$.
- Prove the theorem for permanent polynomial computed by circuits over $\mathbb{Q}$.
- Prove the conjecture.

Open Problems

- Prove the theorem for $s(n) = 2^{\Omega(n)}$.
- Prove the theorem for permanent polynomial computed by circuits over $\mathbb{Q}$.
- Prove the conjecture.

Open Problems

- Prove the theorem for $s(n) = 2^{\Omega(n)}$.
- Prove the theorem for permanent polynomial computed by circuits over $\mathbb{Q}$.
- Prove the conjecture.