Deterministic factorization of constant depth algebraic circuits in subexponential time

Somnath Bhattacharjee
(University of Toronto)

Joint work with Mrinal Kumar (TIFR), Varun Ramanathan (TIFR)

Ramprasad Saptharishi (TIFR), Shubhangi Saraf (UofT)

(submitted)

Outline

Disclaimer: We are *rational* person

Recall High-School Algebra

Factorize the polynomial:

Recall High-School Algebra

Factorize the polynomial:

- $x^2 + 10x + 16$

Recall High-School Algebra

Factorize the polynomial:

- $x^2 + 10x + 16$
- $x^4 + 132x^3 - 17x + 12$

Recall High-School Algebra

Factorize the polynomial:

- $x^2 + 10x + 16$
- $x^4 + 132x^3 - 17x + 12$
- $x^3y^2 + 10x^2y^6 + 10x^2 - 10xy + 19$

Recall High-School Algebra

Factorize the polynomial:

- $x^2 + 10x + 16$
- $x^4 + 132x^3 - 17x + 12$
- $x^3y^2 + 10x^2y^6 + 10x^2 - 10xy + 19$

Fact: Univariate Factorization is Easy [LLL '82]
Multivariate not so sure

Multivariate Factorization Problem

Input: A multivariate polynomial
Output: All the irreducible factors with multiplicities

Multivariate Factorization Problem

Input: A multivariate polynomial
Output: All the irreducible factors with multiplicities

- $O(d^n)$ time algorithm for n variate d degree polynomials

Multivariate Factorization Problem

Input: A multivariate polynomial
Output: All the irreducible factors with multiplicities

- $O(d^n)$ time algorithm for n variate d degree polynomials

- n variate d degree polynomial have d^n possible monomials

Multivariate Factorization Problem

Input: A multivariate polynomial
Output: All the irreducible factors with multiplicities

- $O(d^n)$ time algorithm for n variate d degree polynomials

- n variate d degree polynomial have d^n possible monomials

- Efficient when polynomials are given via monomial representation

Algebraic circuit

Polynomials with exponentially many monomials can be represented by small bits

Algebraic circuit

Polynomials with exponentially many monomials can be represented by small bits

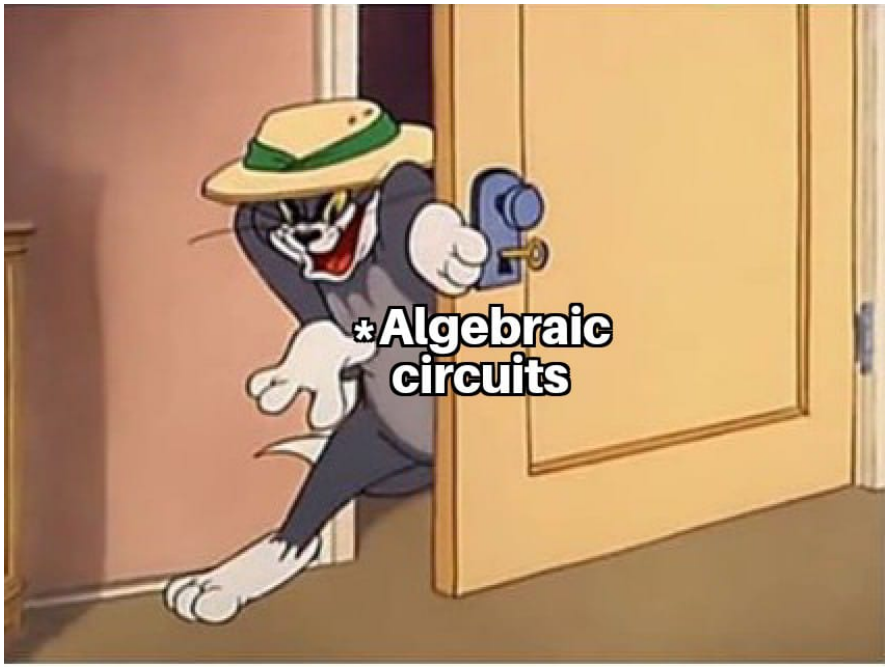
$$(1 + x_1)(1 + x_2) \dots (1 + x_n)$$

Algebraic circuit

Polynomials with exponentially many monomials can be represented by small bits

$$(1 + x_1)(1 + x_2) \dots (1 + x_n)$$

Hence Algebraic Circuit enters the picture



Algebraic Circuits

- Algebraic Circuits are directed acyclic graphs.

Algebraic Circuits

- Algebraic Circuits are directed acyclic graphs.
- Each internal node: $+$ or $\times$ gate.
- Each leaf: constants $\mathbb{Q}$ or variables $\mathbf{X} = \{x_1, \ldots, x_n\}$

Algebraic Circuits

- Algebraic Circuits are directed acyclic graphs.
- Each internal node: $+$ or $\times$ gate.
- Each leaf: constants $\mathbb{Q}$ or variables $\mathbf{X} = \{x_1, \ldots, x_n\}$
- Computes a polynomial in $\mathbb{Q}[\mathbf{X}]$

Algebraic Circuits

- Algebraic Circuits are directed acyclic graphs.
- Each internal node: $+$ or $\times$ gate.
- Each leaf: constants $\mathbb{Q}$ or variables $\mathbf{X} = \{x_1, \ldots, x_n\}$
- Computes a polynomial in $\mathbb{Q}[\mathbf{X}]$
- **Size**: # Edges in the circuit
- **Depth**: length of shortest paths from root to leaf

Polynomial Identity Testing (PIT)

Input: An algebraic circuit $\mathcal{C}$ (size, degree: $poly(n)$)
output: Decide $\mathcal{C} \equiv 0$

Polynomial Identity Testing (PIT)

Input: An algebraic circuit $\mathcal{C}$ (size, degree: $poly(n)$)
output: Decide $\mathcal{C} \equiv 0$

- Randomized one-sided error algorithm in $poly(n)$ time

Polynomial Identity Testing (PIT)

Input: An algebraic circuit $\mathcal{C}$ (size, degree: $poly(n)$)
output: Decide $\mathcal{C} \equiv 0$

- Randomized one-sided error algorithm in $poly(n)$ time

- Evaluate at random point

Polynomial Identity Testing (PIT)

Input: An algebraic circuit $\mathcal{C}$ (size, degree: $poly(n)$)
output: Decide $\mathcal{C} \equiv 0$

- Randomized one-sided error algorithm in $poly(n)$ time

- Evaluate at random point

- n many random bits needed

Algebraic Circuit factorization

Input: An algebraic circuit $\mathcal{C}$ (size, degree: $poly(n)$)
output: Find all irreducible factors

Algebraic Circuit factorization

Input: An algebraic circuit $\mathcal{C}$ (size, degree: $poly(n)$)
output: Find all irreducible factors

- [Kal 86]: Factors of $\mathcal{C}$ has circuit size $poly(n)$

Algebraic Circuit factorization

Input: An algebraic circuit $\mathcal{C}$ (size, degree: $poly(n)$)
output: Find all irreducible factors

- [Kal 86]: Factors of $\mathcal{C}$ has circuit size $poly(n)$
- [Kal 86]: $poly(n)$ time deterministic algorithm to construct them, given oracle access to PIT

Algebraic Circuit factorization

Input: An algebraic circuit $\mathcal{C}$ (size, degree: $poly(n)$)
output: Find all irreducible factors

- [Kal 86]: Factors of $\mathcal{C}$ has circuit size $poly(n)$
- [Kal 86]: $poly(n)$ time deterministic algorithm to construct them, given oracle access to PIT

Polynomial Factorization problem has applications in:

- Coding Theory
- Pseudorandomness
- Cryptography
- Graph algorithms

Constant-Depth Algebraic Circuit factorization

- Randomized Algorithm for Polynomial factorization with n many random bits

Constant-Depth Algebraic Circuit factorization

- Randomized Algorithm for Polynomial factorization with n many random bits (Same as PIT)

Constant-Depth Algebraic Circuit factorization

- Randomized Algorithm for Polynomial factorization with n many random bits (Same as PIT)
- [LST 21]: sub-exponential PIT for constant depth circuits

Constant-Depth Algebraic Circuit factorization

- Randomized Algorithm for Polynomial factorization with n many random bits (Same as PIT)
- [LST 21]: sub-exponential PIT for constant depth circuits
- Can we factor constant depth circuits in sub-exp?

Constant-Depth Algebraic Circuit factorization

- Randomized Algorithm for Polynomial factorization with n many random bits (Same as PIT)
- [LST 21]: sub-exponential PIT for constant depth circuits
- Can we factor constant depth circuits in sub-exp?
- [KRS 23], [DST24], [KRSV 24] made progresses towards this

Constant-Depth Algebraic Circuit factorization

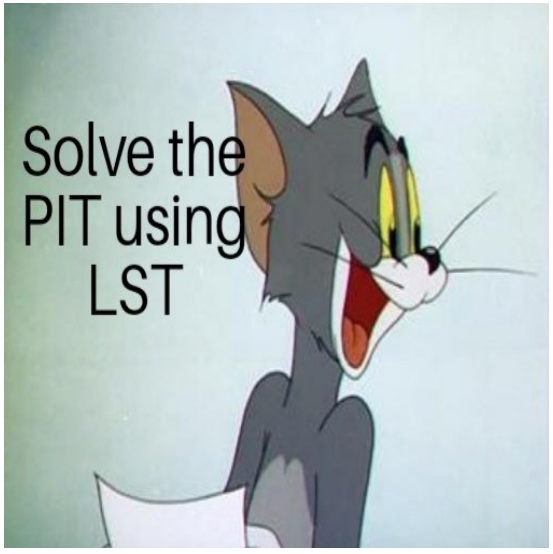
- Randomized Algorithm for Polynomial factorization with n many random bits (Same as PIT)
- [LST 21]: sub-exponential PIT for constant depth circuits
- Can we factor constant depth circuits in sub-exp?
- [KRS 23], [DST24], [KRSV 24] made progresses towards this
- Finally we answered the question: Yes, we can!

Why so difficult

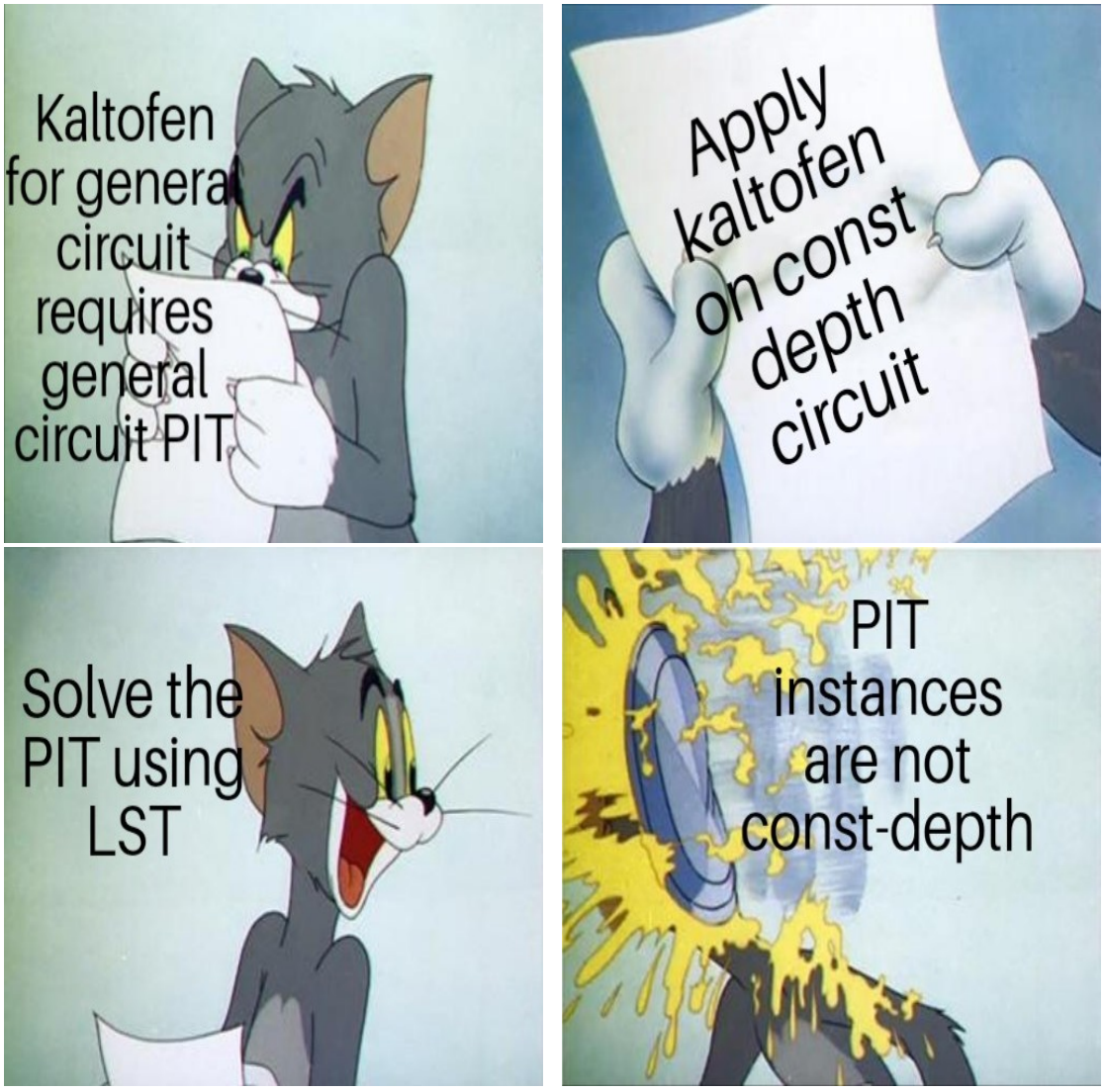
Why so difficult

Why so difficult

Why so difficult



Why so difficult



Newton Iteration/ Hensel Lifting destroyes the fun

Somehow we did it

Main Result

Theorem

Given constant $\varepsilon > 0$, *we can factor constant-depth circuit* $\mathcal{C}(x_1, \ldots, x_n)$ *of size* $poly(n)$ *in* $\tilde{O}(n^{n^{\varepsilon}})$

Main Result

Theorem

Given constant $\varepsilon > 0$, *we can factor constant-depth circuit* $\mathcal{C}(x_1, \ldots, x_n)$ *of size* $poly(n)$ *in* $\tilde{O}(n^{n^{\varepsilon}})$

Note **Brute-force** *will take* $\tilde{O}(n^n)$ *time*

Main Result

Theorem

Given constant $\varepsilon > 0$, *we can factor constant-depth circuit* $\mathcal{C}(x_1, \ldots, x_n)$ *of size* $poly(n)$ *in* $\tilde{O}(n^{n^\varepsilon})$

Note **Brute-force** *will take* $\tilde{O}(n^n)$ *time*

Key Step: Solve the PIT instances in Kaltofen's Algorithm

Power series Factors VS univariate factors

Input $\mathcal{C}(x_1, \ldots, x_n, y)$ is monic in y and square-free

Power series Factors VS univariate factors

Input $\mathcal{C}(x_1, \ldots, x_n, y)$ is monic in y and square-free

Theorem

Hensel: *$\mathcal{C}(\mathbf{X}, y)$ facotrizes completely over $\mathbb{Q}[[\mathbf{X}]][y]$*

Power series Factors VS univariate factors

Input $\mathcal{C}(x_1, \ldots, x_n, y)$ is monic in y and square-free

Theorem

Hensel: *$\mathcal{C}(\mathbf{X}, y)$ facotrizes completely over $\mathbb{Q}[[\mathbf{X}]][y]$*

$$\mathcal{C}(\mathbf{X}, y) = (y - \varphi_1(\mathbf{X})) \ldots (y - \varphi_d(\mathbf{X})) \quad (\varphi_i\text{: power series in } \mathbf{X})$$
$$\mathcal{C}(0, y) = (y - u_1) \ldots (y - u_d)$$

Power series Factors VS univariate factors

Input $\mathcal{C}(x_1, \ldots, x_n, y)$ is monic in y and square-free

Theorem

Hensel: *$\mathcal{C}(\mathbf{X}, y)$ facotrizes completely over $\mathbb{Q}[[\mathbf{X}]][y]$*

$$\mathcal{C}(\mathbf{X}, y) = (y - \varphi_1(\mathbf{X})) \ldots (y - \varphi_d(\mathbf{X})) \quad (\varphi_i\text{: power series in } \mathbf{X})$$
$$\mathcal{C}(0, y) = (y - u_1) \ldots (y - u_d)$$

u is the constant in $\varphi(\mathbf{X})$

Newton Iteration

$$\mathcal{C}(\mathbf{X}, y) = (y - \varphi_1(\mathbf{X})) \dots (y - \varphi_d(\mathbf{X}))$$
$$\mathcal{C}(0, y) = (y - u_1) \dots (y - u_d)$$

Newton Iteration

$$\mathcal{C}(\mathbf{X}, y) = (y - \varphi_1(\mathbf{X})) \dots (y - \varphi_d(\mathbf{X}))$$
$$\mathcal{C}(0, y) = (y - u_1) \dots (y - u_d)$$

- $\varphi^{(r)}$: $\leq r$ degree part of φ

Newton Iteration

$$\mathcal{C}(\mathbf{X}, y) = (y - \varphi_1(\mathbf{X})) \dots (y - \varphi_d(\mathbf{X}))$$
$$\mathcal{C}(0, y) = (y - u_1) \dots (y - u_d)$$

- $\varphi^{(r)}$: $\leq r$ degree part of φ
- $\varphi^{(0)} = u$

Newton Iteration

$$\mathcal{C}(\mathbf{X}, y) = (y - \varphi_1(\mathbf{X})) \dots (y - \varphi_d(\mathbf{X}))$$
$$\mathcal{C}(0, y) = (y - u_1) \dots (y - u_d)$$

- $\varphi^{(r)}$: $\leq r$ degree part of φ
- $\varphi^{(0)} = u$
- $\varphi^{(r+1)}$ can be computed from $\varphi^{(0)}, \dots, \varphi^{(r-1)}, \varphi^{(r)}, \mathcal{C}$

Newton Iteration

$$\mathcal{C}(\mathbf{X}, y) = (y - \varphi_1(\mathbf{X}))\dots(y - \varphi_d(\mathbf{X}))$$
$$\mathcal{C}(0, y) = (y - u_1)\dots(y - u_d)$$

- $\varphi^{(r)}$: $\leq r$ degree part of φ
- $\varphi^{(0)} = u$
- $\varphi^{(r+1)}$ can be computed from $\varphi^{(0)}, \dots, \varphi^{(r-1)}, \varphi^{(r)}, \mathcal{C}$
- $\varphi^{(r)}$ has $O(r)$-depth circuit

Kaltofen's Algorithm

Kaltofen's Algorithm

- Factorize $\mathcal{C}(0, y)$

Kaltofen's Algorithm

- Factorize $\mathcal{C}(0, y)$
- from univariate root u_i, compute φ_i

Kaltofen's Algorithm

- Factorize $\mathcal{C}(0, y)$
- from univariate root u_i, compute φ_i

How to find the actual factor?

Kaltofen's Algorithm

- Factorize $\mathcal{C}(0, y)$
- from univariate root u_i, compute φ_i

How to find the actual factor?

- That's where PIT comes in

Finding a true root

Want to check: φ is a true root?

Finding a true root

Want to check: φ is a true root?

$$\Updownarrow$$

$$\tilde{\mathcal{C}}(\mathbf{X}) := \mathcal{C}(\mathbf{X}, \varphi^{(d)}(\mathbf{X})) = 0?$$

Finding a true root

Want to check: φ is a true root?

$$\Updownarrow$$

$$\tilde{C}(\mathbf{X}) := C(\mathbf{X}, \varphi^{(d)}(\mathbf{X})) = 0?$$

$\tilde{C}$ is **NOT** constant-depth,

Finding a true root

Want to check: φ is a true root?

$$\Updownarrow$$

$$\tilde{C}(\mathbf{X}) := C(\mathbf{X}, \varphi^{(d)}(\mathbf{X})) = 0?$$

$\tilde{C}$ is **NOT** constant-depth, but *constant-depth like*

Highlevel idea

- **So far:** we need PIT to do factorization

Highlevel idea

- **So far:** we need PIT to do factorization
- to do the PIT, we need different factorization result

Highlevel idea

- **So far:** we need PIT to do factorization
- to do the PIT, we need different factorization result
- Don't get confused :)

Constant-depth PIT idea

- [CKS 18]: Low degree roots of constant depth circuit has small constant-depth circuit

Constant-depth PIT idea

- [CKS 18]: Low degree roots of constant depth circuit has small constant-depth circuit
- Let $f(z_1, \ldots, z_m)$ does **not** have small constant-depth circuit ($m < n$)

Constant-depth PIT idea

- [CKS 18]: Low degree roots of constant depth circuit has small constant-depth circuit
- Let $f(z_1, \ldots, z_m)$ does **not** have small constant-depth circuit ($m < n$)

$$\mathcal{C} \circ f(z_1, \ldots, z_m) \neq 0$$

Constant-depth PIT idea

- [CKS 18]: Low degree roots of constant depth circuit has small constant-depth circuit
- Let $f(z_1, \ldots, z_m)$ does **not** have small constant-depth circuit ($m < n$)

$$\mathcal{C} \circ f(z_1, \ldots, z_m) \neq 0$$

Else f is a root of $\mathcal{C}$, has small circuit

Constant-depth PIT idea

- [CKS 18]: Low degree roots of constant depth circuit has small constant-depth circuit
- Let $f(z_1, \ldots, z_m)$ does **not** have small constant-depth circuit ($m < n$)

$$\mathcal{C} \circ f(z_1, \ldots, z_m) \neq 0$$

  Else f is a root of $\mathcal{C}$, has small circuit

- [LST 21]: Gave an explicit f

Constant-depth PIT idea

- [CKS 18]: Low degree roots of constant depth circuit has small constant-depth circuit
- Let $f(z_1, \ldots, z_m)$ does **not** have small constant-depth circuit ($m < n$)

$$\mathcal{C} \circ f(z_1, \ldots, z_m) \neq 0$$

  Else f is a root of $\mathcal{C}$, has small circuit
- [LST 21]: Gave an explicit f
- [KI 04]: Gave an explicit variable reduction using such f which preserves non-zeroness.

Constant-depth-like PIT idea

Key lemma

Low degree roots of constant-depth like circuits also have small constant depth circuit

Highlevel idea: Newton iteration (with quadratic convergence)

Same [KI] variable reduction will work

Field dependency

- This algorithm is efficient in terms of bit-complexity
- Works for any field with $char > \omega(d)$ or $= 0$ and univariate factorization is known

Open questions

- Can we avoid the dependency on the hardness result? i.e., can we use the constant depth PIT result black-box to solve factorization?
- Are constant depth circuits closed under factorization?
- Can we do something better for sparse polynomials?