On the Hardness of PosSLP

Peter Bürgisser[1] Gorav Jindal[2]

January 8, 2024

SODA 2024, Alexandria, Virginia, USA

[1]Institut für Mathematik, Technische Universität Berlin, Germany

[2]Max Planck Institute for Software Systems, Saarbrücken, Germany

Table of Contents

1 Motivation

2 PosSLP

3 Conditional Hardness of PosSLP

Motivation: Numerical stable algorithms

- An algorithm (represented as function f) with $f\colon \mathbb{R}^n \to \mathbb{R}$
- On an input $x = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$, compute $f(x)$

Motivation: Numerical stable algorithms

- An algorithm (represented as function f) with $f\colon \mathbb{R}^n \to \mathbb{R}$
- On an input $x = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$, compute $f(x)$
- Exact computation of $f(x)$ may not be possible because:

ACM-SIAM Symposium on Discrete Algorithms

Motivation: Numerical stable algorithms

- An algorithm (represented as function f) with $f\colon \mathbb{R}^n \to \mathbb{R}$
- On an input $x = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$, compute $f(x)$
- Exact computation of $f(x)$ may not be possible because:
  - f may evaluate to irrational numbers

SIAM 2024 ACM-SIAM Symposium on Discrete Algorithms

Motivation: Numerical stable algorithms

- An algorithm (represented as function f) with $f\colon \mathbb{R}^n \to \mathbb{R}$
- On an input $x = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$, compute $f(x)$
- Exact computation of $f(x)$ may not be possible because:
  - f may evaluate to irrational numbers
  - Only an approximation $\tilde{x}$ of x might be known in practice

Motivation: Numerical stable algorithms

- An algorithm (represented as function f) with $f\colon \mathbb{R}^n \to \mathbb{R}$
- On an input $x = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$, compute $f(x)$
- Exact computation of $f(x)$ may not be possible because:
  - f may evaluate to irrational numbers
  - Only an approximation $\tilde{x}$ of x might be known in practice
  - Exact computation $f(x)$ may be computationally expensive

Motivation: Numerical stable algorithms

- An algorithm (represented as function f) with $f\colon \mathbb{R}^n \to \mathbb{R}$
- On an input $x = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$, compute $f(x)$
- Exact computation of $f(x)$ may not be possible because:
  - f may evaluate to irrational numbers
  - Only an approximation $\tilde{x}$ of x might be known in practice
  - Exact computation $f(x)$ may be computationally expensive
- Can we efficiently approximate $f(x)$?

Motivation: Numerical stable algorithms

- An algorithm (represented as function f) with $f\colon \mathbb{R}^n \to \mathbb{R}$
- On an input $x = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$, compute $f(x)$
- Exact computation of $f(x)$ may not be possible because:
  - f may evaluate to irrational numbers
  - Only an approximation $\tilde{x}$ of x might be known in practice
  - Exact computation $f(x)$ may be computationally expensive
- Can we efficiently approximate $f(x)$?
- It is reasonable to assume that f can be approximated using polynomials

Arithmetic circuits and SLPs

ACM-SIAM Symposium on Discrete Algorithms

Arithmetic circuits and SLPs

Definition (Arithmetic circuit)

An arithmetic circuit is a directed acyclic graph whose inputs are constants $0, 1$ or indeterminates $x_1, x_2, \ldots, x_n$. Internal nodes are operations $+, -, \times, \div$.

Arithmetic circuits and SLPs

Definition (Arithmetic circuit)

An arithmetic circuit is a directed acyclic graph whose inputs are constants $0, 1$ or indeterminates $x_1, x_2, \ldots, x_n$. Internal nodes are operations $+, -, \times, \div$.

- Each arithmetic circuit computes a rational function $\frac{f}{g}$ with $f, g \in \mathbb{Z}[x_1, x_2, \ldots, x_n]$
- Size = Number of nodes

Arithmetic circuits and SLPs

Definition (Arithmetic circuit)

An arithmetic circuit is a directed acyclic graph whose inputs are constants $0, 1$ or indeterminates $x_1, x_2, \ldots, x_n$. Internal nodes are operations $+, -, \times, \div$.

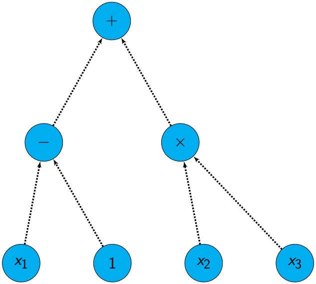
- Each arithmetic circuit computes a rational function $\frac{f}{g}$ with $f, g \in \mathbb{Z}[x_1, x_2, \ldots, x_n]$
- Size $=$ Number of nodes

Definition (SLP)

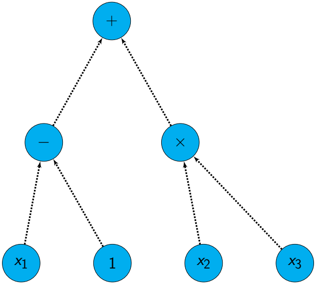
A straight-line program (SLP) is a sequence of instructions for evaluation of an arithmetic circuit.

- SLPs and arithmetic circuits are used interchangeably.

Example



Example



- This circuit computes the polynomial $(x_1 - 1) + x_2 x_3$

Floating point representations

- For any non-zero $u \in \mathbb{R}$, there exists unique $u' \in \mathbb{R}, m \in \mathbb{Z}$ with $\frac{1}{2} \leq |u'| < 1$ such that $u = u'2^m$

ACM-SIAM Symposium on Discrete Algorithms

Floating point representations

- For any non-zero $u \in \mathbb{R}$, there exists unique $u' \in \mathbb{R}, m \in \mathbb{Z}$ with $\frac{1}{2} \leq |u'| < 1$ such that $u = u'2^m$
- For $k \in \mathbb{N}$, approximate u' by a v such that $|v - u'| \leq 2^{-(k+1)}$

Floating point representations

- For any non-zero $u \in \mathbb{R}$, there exists unique $u' \in \mathbb{R}, m \in \mathbb{Z}$ with $\frac{1}{2} \leq |u'| < 1$ such that $u = u'2^m$
- For $k \in \mathbb{N}$, approximate u' by a v such that $|v - u'| \leq 2^{-(k+1)}$
- This pair (v, m) is a floating point approximation of u with k significant bits

ACM-SIAM Symposium on Discrete Algorithms

Task of a numerical analyst

- Given a function $f : \mathbb{R}^n \to \mathbb{R}$ and $x = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$ as inputs, approximate $f(x)$

SIAM 2024 | ACM-SIAM Symposium on Discrete Algorithms

Task of a numerical analyst

- Given a function $f : \mathbb{R}^n \to \mathbb{R}$ and $x = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$ as inputs, approximate $f(x)$
- There is a method to compute or approximate f

Task of a numerical analyst

- Given a function $f : \mathbb{R}^n \to \mathbb{R}$ and $x = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$ as inputs, approximate $f(x)$
- There is a method to compute or approximate f
- Assume f can be computed using an SLP

Task of a numerical analyst

- Given a function $f : \mathbb{R}^n \to \mathbb{R}$ and $x = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$ as inputs, approximate $f(x)$
- There is a method to compute or approximate f
- Assume f can be computed using an SLP

Problem (Generic task of numerical computation (GTNC))

Given a SLP P with indeterminates $x_1, x_2, \ldots, x_n$, floating point numbers $a_1, a_2, \ldots, a_n$ and an integer k in unary, compute a floating point approximation of $P(a_1, a_2, \ldots, a_n)$ with k significant bits.

ACM-SIAM Symposium on Discrete Algorithms

Table of Contents

1 Motivation

2 PosSLP

3 Conditional Hardness of PosSLP

PosSLP

- Motivation: To characterize the complexity of numerical analysis (GTNC)

PosSLP

- Motivation: To characterize the complexity of numerical analysis (GTNC)

Definition (PosSLP)

Given a division-free SLP P without indeterminates, decide if the integer N computed by P is positive.

PosSLP

- Motivation: To characterize the complexity of numerical analysis (GTNC)

Definition (PosSLP)

Given a division-free SLP P without indeterminates, decide if the integer N computed by P is positive.

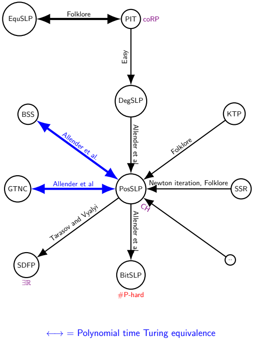
- Such an SLP P is sequence of integers $(b_0, b_1, b_2, \ldots, b_\ell)$ with $b_0 = 1$ and for all $1 \leq i \leq \ell$, $b_i = b_j \circ_i b_k$, where $\circ_i \in \{+, -, *\}$ and $j, k < i$
- Integer computed by P is b_ℓ, Size of P is ℓ

Connection to numerical analysis

Theorem ((Allender et al. 2006))

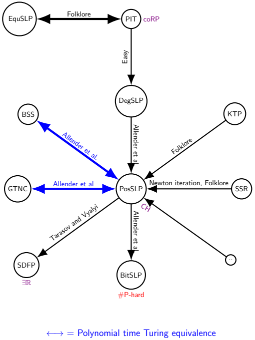
GTNC *is polynomial time Turing equivalent to* PosSLP.

Complexity landscape of PosSLP



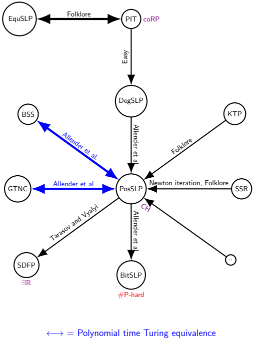
- EquSLP: Given a division-free SLP computing $N \in \mathbb{Z}$, decide if $N = 0$

Complexity landscape of PosSLP



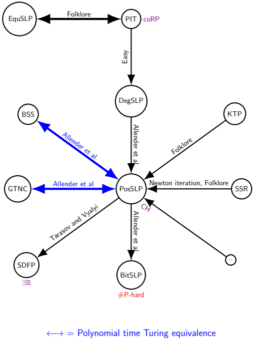
- EquSLP: Given a division-free SLP computing $N \in \mathbb{Z}$, decide if $N = 0$
- PIT: Given a division-free arithmetic circuit computing a polynomial $f \in \mathbb{Z}[x_1, x_2, \ldots, x_n]$, decide if $f = 0$

Complexity landscape of PosSLP



- EquSLP: Given a division-free SLP computing $N \in \mathbb{Z}$, decide if $N = 0$
- PIT: Given a division-free arithmetic circuit computing a polynomial $f \in \mathbb{Z}[x_1, x_2, \ldots, x_n]$, decide if $f = 0$
- DegSLP: Given a division-free arithmetic circuit computing a polynomial $f \in \mathbb{Z}[x_1, x_2, \ldots, x_n]$ and $d \in \mathbb{N}$, decide if $\deg(f) \leq d$

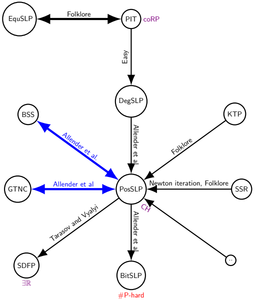
Complexity landscape of PosSLP



- EquSLP: Given a division-free SLP computing $N \in \mathbb{Z}$, decide if $N = 0$
- PIT: Given a division-free arithmetic circuit computing a polynomial $f \in \mathbb{Z}[x_1, x_2, \ldots, x_n]$, decide if $f = 0$
- DegSLP: Given a division-free arithmetic circuit computing a polynomial $f \in \mathbb{Z}[x_1, x_2, \ldots, x_n]$ and $d \in \mathbb{N}$, decide if $\deg(f) \leq d$
- BitSLP: Given a division-free SLP computing $N \in \mathbb{Z}$ and $i \in \mathbb{N}$, decide if i^{th} bit of N is 1

ACM-SIAM Symposium on Discrete Algorithms

Complexity landscape of PosSLP

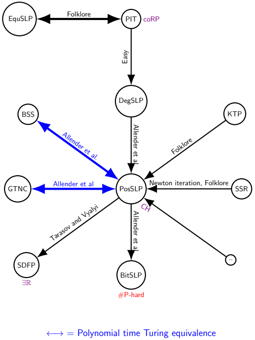


$\longleftrightarrow$ = Polynomial time Turing equivalence

- EquSLP: Given a division-free SLP computing $N \in \mathbb{Z}$, decide if $N = 0$
- PIT: Given a division-free arithmetic circuit computing a polynomial $f \in \mathbb{Z}[x_1, x_2, \ldots, x_n]$, decide if $f = 0$
- DegSLP: Given a division-free arithmetic circuit computing a polynomial $f \in \mathbb{Z}[x_1, x_2, \ldots, x_n]$ and $d \in \mathbb{N}$, decide if $\deg(f) \leq d$
- BitSLP: Given a division-free SLP computing $N \in \mathbb{Z}$ and $i \in \mathbb{N}$, decide if i^{th} bit of N is 1
- Semidefinite feasibility problem (SDFP): Given an affine subspace of matrices, decide if it contains a positive semidefinite matrix

siam 2024 ACM-SIAM Symposium on Discrete Algorithms

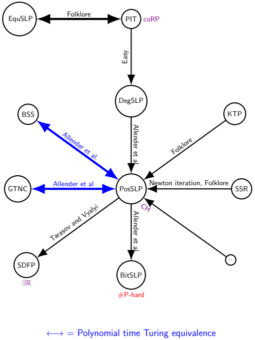
Complexity landscape of PosSLP



- EquSLP: Given a division-free SLP computing $N \in \mathbb{Z}$, decide if $N = 0$
- PIT: Given a division-free arithmetic circuit computing a polynomial $f \in \mathbb{Z}[x_1, x_2, \ldots, x_n]$, decide if $f = 0$
- DegSLP: Given a division-free arithmetic circuit computing a polynomial $f \in \mathbb{Z}[x_1, x_2, \ldots, x_n]$ and $d \in \mathbb{N}$, decide if $\deg(f) \leq d$
- BitSLP: Given a division-free SLP computing $N \in \mathbb{Z}$ and $i \in \mathbb{N}$, decide if i^{th} bit of N is 1
- Semidefinite feasibility problem (SDFP): Given an affine subspace of matrices, decide if it contains a positive semidefinite matrix
- BSS: Blum, Shub, and Smale model

ACM-SIAM Symposium on Discrete Algorithms

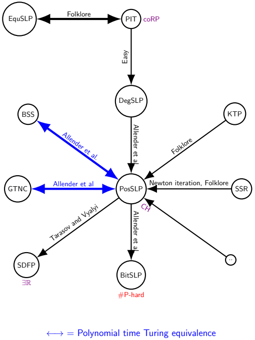
Complexity landscape of PosSLP



- EquSLP: Given a division-free SLP computing $N \in \mathbb{Z}$, decide if $N = 0$
- PIT: Given a division-free arithmetic circuit computing a polynomial $f \in \mathbb{Z}[x_1, x_2, \ldots, x_n]$, decide if $f = 0$
- DegSLP: Given a division-free arithmetic circuit computing a polynomial $f \in \mathbb{Z}[x_1, x_2, \ldots, x_n]$ and $d \in \mathbb{N}$, decide if $\deg(f) \leq d$
- BitSLP: Given a division-free SLP computing $N \in \mathbb{Z}$ and $i \in \mathbb{N}$, decide if i^{th} bit of N is 1
- Semidefinite feasibility problem (SDFP): Given an affine subspace of matrices, decide if it contains a positive semidefinite matrix
- BSS: Blum, Shub, and Smale model
- SSR: Sum of Square roots problem
- KTP: Koiran's trinomial sign problem (Koiran 2019)

ACM-SIAM Symposium on Discrete Algorithms

Complexity landscape of PosSLP



- EquSLP: Given a division-free SLP computing $N \in \mathbb{Z}$, decide if $N = 0$
- PIT: Given a division-free arithmetic circuit computing a polynomial $f \in \mathbb{Z}[x_1, x_2, \ldots, x_n]$, decide if $f = 0$
- DegSLP: Given a division-free arithmetic circuit computing a polynomial $f \in \mathbb{Z}[x_1, x_2, \ldots, x_n]$ and $d \in \mathbb{N}$, decide if $\deg(f) \leq d$
- BitSLP: Given a division-free SLP computing $N \in \mathbb{Z}$ and $i \in \mathbb{N}$, decide if i^{th} bit of N is 1
- Semidefinite feasibility problem (SDFP): Given an affine subspace of matrices, decide if it contains a positive semidefinite matrix
- BSS: Blum, Shub, and Smale model
- SSR: Sum of Square roots problem
- KTP: Koiran's trinomial sign problem (Koiran 2019)
- CH: Counting hierarchy
- $\exists\mathbb{R}$: decide if a given semialgebraic set is non empty

Upper bounds for PosSLP

Theorem ((Allender et al. 2006))

PosSLP $\in$ CH, *here* CH *is the counting hierarchy.*

Upper bounds for PosSLP

Theorem ((Allender et al. 2006))

PosSLP $\in$ CH, *here* CH *is the counting hierarchy.*

- Another approach: for $n \in \mathbb{N}$,
  - $\tau(n) :=$ size of the smallest SLP which computes n
  - $\tau_+(n) :=$ size of the smallest subtraction free SLP which computes n

ACM-SIAM Symposium on Discrete Algorithms

Upper bounds for PosSLP

Theorem ((Allender et al. 2006))

PosSLP $\in$ CH, *here* CH *is the counting hierarchy.*

- Another approach: for $n \in \mathbb{N}$,
  - $\tau(n) :=$ size of the smallest SLP which computes n
  - $\tau_+(n) :=$ size of the smallest subtraction free SLP which computes n
- If $\tau_+(n) \leq \text{poly}(\tau(n))$ then PosSLP $\in$ PH (Jindal and Saranurak 2012)

Upper bounds for PosSLP

Theorem ((Allender et al. 2006))

PosSLP $\in$ CH, *here* CH *is the counting hierarchy.*

- Another approach: for $n \in \mathbb{N}$,
  - $\tau(n) :=$ size of the smallest SLP which computes n
  - $\tau_+(n) :=$ size of the smallest subtraction free SLP which computes n
- If $\tau_+(n) \leq \text{poly}(\tau(n))$ then PosSLP $\in$ PH (Jindal and Saranurak 2012)
- There exist integer sequences where $\tau_+(n) > \tau(n)$ (Jindal and Saranurak 2012)

ACM-SIAM Symposium on Discrete Algorithms

Lower bounds for PosSLP

- Unfortunately, nontrivial lower bounds for PosSLP remain unknown

Lower bounds for PosSLP

- Unfortunately, nontrivial lower bounds for PosSLP remain unknown

Theorem (This paper)

If a constructive variant of the **radical conjecture** *is true and* PosSLP $\in$ BPP *then* NP $\subseteq$ BPP.

Table of Contents

1 Motivation

2 PosSLP

3 Conditional Hardness of PosSLP

siam 2024 ACM-SIAM Symposium on Discrete Algorithms

Existence of real roots

- RealRootSLP: Given an arithmetic circuit computing a univariate polynomial $f \in \mathbb{Z}[x]$, decide if f has a real root

Existence of real roots

- RealRootSLP: Given an arithmetic circuit computing a univariate polynomial $f \in \mathbb{Z}[x]$, decide if f has a real root
- 3SAT$\leq_{\mathrm{P}}$ RealRootSLP, hence RealRootSLP is NP-hard (Perrucci and Sabia 2007)

Existence of real roots

- RealRootSLP: Given an arithmetic circuit computing a univariate polynomial $f \in \mathbb{Z}[x]$, decide if f has a real root
- 3SAT$\leq_{\mathrm{P}}$ RealRootSLP, hence RealRootSLP is NP-hard (Perrucci and Sabia 2007)
  - 3SAT formula $\phi \longrightarrow$ An arithmetic circuit computing a polynomial f_ϕ such that ϕ is satisfiable iff f_ϕ has a real root

Existence of real roots

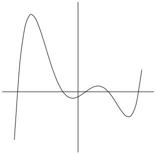
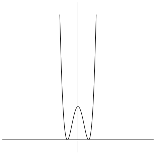
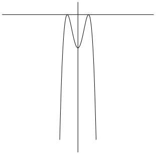
- RealRootSLP: Given an arithmetic circuit computing a univariate polynomial $f \in \mathbb{Z}[x]$, decide if f has a real root
- 3SAT$\leq_{\mathrm{P}}$ RealRootSLP, hence RealRootSLP is NP-hard (Perrucci and Sabia 2007)
  - 3SAT formula $\phi \longrightarrow$ An arithmetic circuit computing a polynomial f_ϕ such that ϕ is satisfiable iff f_ϕ has a real root
  - All the real roots (if any) of f_ϕ are in $(-1, 1)$

Idea for NP-hardness of PosSLP

- If ϕ is satisfiable then f_ϕ "should" look like:

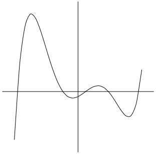
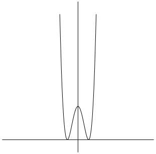
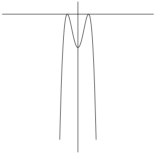
Idea for NP-hardness of PosSLP

- If ϕ is satisfiable then f_ϕ "should" look like:

Idea for NP-hardness of PosSLP

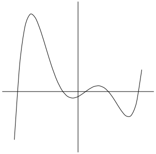
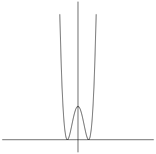
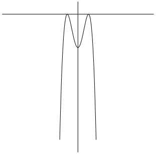
- If ϕ is satisfiable then f_ϕ “should” look like:

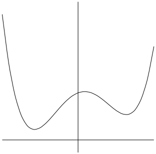
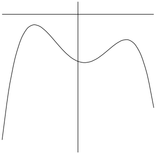
- If ϕ is not satisfiable then f_ϕ “should” look like:

Idea for NP-hardness of PosSLP

- If ϕ is satisfiable then f_ϕ "should" look like:

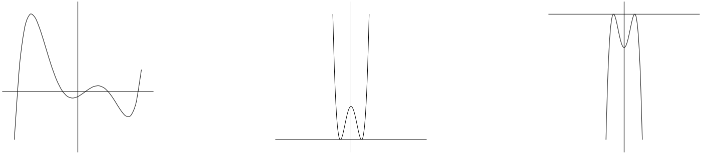




- If ϕ is not satisfiable then f_ϕ "should" look like:

Idea for NP-hardness of PosSLP

- If ϕ is satisfiable then f_ϕ "should" look like:



- If ϕ is not satisfiable then f_ϕ "should" look like:

- Pick a random rational q in $(-1, 1)$, check if $f_\phi(1)$ and $f_\phi(q)$ have different signs, i.e, $(-1)f_\phi(1)f_\phi(q) > 0$

ACM-SIAM Symposium on Discrete Algorithms

Challenges

- f_ϕ only changes sign on real roots of odd multiplicity

Challenges

- f_ϕ only changes sign on real roots of odd multiplicity
- What if f_ϕ has roots only of even multiplicity? $\rightsquigarrow$ **Radical conjecture**

Challenges

- f_ϕ only changes sign on real roots of odd multiplicity
- What if f_ϕ has roots only of even multiplicity? $\rightsquigarrow$ **Radical conjecture**
- Even if f_ϕ has roots only of odd multiplicity, how likely is it that $f_\phi(1)$ and $f_\phi(q)$ have different signs? $\rightsquigarrow$ **UniqueSAT**

Radical conjecture

- $\tau(f) =$ size of the smallest arithmetic circuit computing f

Radical conjecture

- $\tau(f) =$ size of the smallest arithmetic circuit computing f
- $\text{rad}(f) =$ Radical of f, i.e, the square free part of f

Radical conjecture

- $\tau(f) =$ size of the smallest arithmetic circuit computing f
- $\text{rad}(f) =$ Radical of f, i.e, the square free part of f
- $\text{rad}(f)$ has only simple roots and has all the roots of f

Radical conjecture

- $\tau(f) =$ size of the smallest arithmetic circuit computing f
- $\mathrm{rad}(f) =$ Radical of f, i.e, the square free part of f
- $\mathrm{rad}(f)$ has only simple roots and has all the roots of f

Conjecture ((Dutta, Saxena, and Sinhababu 2022), Radical conjecture)

For univariate $f \in \mathbb{Z}[x]$, $\tau(\mathrm{rad}(f)) \leq \mathrm{poly}(\tau(f))$.

- It implies $\tau(\mathrm{rad}(f_\phi)) \leq \mathrm{poly}(n)$, $n =$ number of literals in ϕ

SIAM 2024 ACM-SIAM Symposium on Discrete Algorithms

UniqueSAT

- Under randomized polynomial time reductions, ϕ can be assumed to have a unique satisfying assignment (Valiant and Vazirani 1986)

UniqueSAT

- Under randomized polynomial time reductions, ϕ can be assumed to have a unique satisfying assignment (Valiant and Vazirani 1986)

Theorem (This paper)

If ϕ has a unique satisfying assignment then $\mathrm{rad}(f_\phi)(1)$ and $\mathrm{rad}(f_\phi)(q)$ have different signs with probability at least $\frac{1}{4\pi}$, where q is a random rational in $(-1, 1)$.

ACM-SIAM Symposium on Discrete Algorithms

Lower Bound

Lower Bound

Theorem (This paper)

If a constructive variant of the radical conjecture is true and PosSLP $\in$ BPP *then* NP $\subseteq$ BPP.

Future research directions

- Radical conjecture is a strong assumption, can we do without it?

Future research directions

- Radical conjecture is a strong assumption, can we do without it?
- Special cases of PosSLP:
  - Koiran's trinomial sign problem (Koiran 2019)
  - Sum of square roots problem
  - Decide if $a^n + b^n - c^n > 0$
  - Many more.....

Thanks for your attention! Any questions?

Literature I

Allender, E. et al. (2006). “On the complexity of numerical analysis”. In: *21st Annual IEEE Conference on Computational Complexity (CCC'06)*, 9 pp.–339. DOI: 10.1109/CCC.2006.30.

Dutta, Pranjal, Nitin Saxena, and Amit Sinhababu (June 2022). “Discovering the Roots: Uniform Closure Results for Algebraic Classes Under Factoring”. In: *J. ACM* 69.3. ISSN: 0004-5411. DOI: 10.1145/3510359. URL: https://doi.org/10.1145/3510359.

Jindal, Gorav and Thatchaphol Saranurak (2012). “Subtraction makes computing integers faster”. In: *CoRR* abs/1212.2549. arXiv: 1212.2549. URL: http://arxiv.org/abs/1212.2549.

Koiran, Pascal (2019). “Root separation for trinomials”. In: *Journal of Symbolic Computation* 95, pp. 151–161. ISSN: 0747-7171. DOI: https://doi.org/10.1016/j.jsc.2019.02.004. URL: https://www.sciencedirect.com/science/article/pii/S074771711930015X.

Literature II

Perrucci, Daniel and Juan Sabia (2007). "Real roots of univariate polynomials and straight line programs". In: *Journal of Discrete Algorithms* 5.3. Selected papers from Ad Hoc Now 2005, pp. 471–478. ISSN: 1570-8667. DOI: https://doi.org/10.1016/j.jda.2006.10.002. URL: https://www.sciencedirect.com/science/article/pii/S1570866706000840.

Valiant, L.G. and V.V. Vazirani (1986). "NP is as easy as detecting unique solutions". In: *Theoretical Computer Science* 47, pp. 85–93. ISSN: 0304-3975. DOI: https://doi.org/10.1016/0304-3975(86)90135-0. URL: https://www.sciencedirect.com/science/article/pii/0304397586901350.