Faster Satisfiability Algorithms for Systems of Polynomial Equations over Finite Fields ~~and ACC^0[p]~~

Suguru TAMAKI *Kyoto University*

Joint work with:

Daniel Lokshtanov, Ramamohan Paturi, Ryan Williams

Satisfiability Lower Bounds and Tight Results for Parameterized and Exponential-Time Algorithms
November 6, 2015, Simons Institute, Berkeley, CA

Systems of Polynomial Equations

have been studied for more than 300 years:

Resultant and Elimination Theory were used by

関 孝和 1642-1708

Étienne Bézout 1730-1783

(Pictures from Wikipedia)

Our Problem: SysPolyEqs(q)

Systems of Polynomial Equations over GF[q]

Input:

GF[q] polynomials $p_1, p_2, \ldots, p_m$

in formal variables $x_1, x_2, \ldots, x_n$

e.g. $q = 3, p_1 = 2x_1^2\, x_2^2\, x_3 + x_3^2\, x_4, p_2 = x_1 x_2 + x_2^2 + 1$

Task:

find a satisfying assignment $a \in \mathrm{GF}[q]^n$

i.e. $p_1(a) = p_2(a) = \cdots = p_m(a) = 0$ holds

e.g. $(x_1, x_2, x_3, x_4) = (2,2,1,1)$

(#SysPolyEqs(q) denotes the counting version)

Complexity of SysPolyEqs(q)

- P if each polynomial has degree 1 (linear equations)
- NP-complete if each polynomial has degree ≤ 2
- Satisfying a $2^{1-d} - 2^{1-2d} + \varepsilon$ fraction of equations is NP-hard on satisfiable instances when $q = 2$, degree $\leq d$ [Hastad'11]
- Best worst-case upper bound: $q^n \times \mathrm{poly}(\text{input-size})$ (even if $q = 2$, degree ≤ 2)

Input:
GF[q] polynomials $p_1, p_2, \dots, p_m$
in formal variables $x_1, x_2, \dots, x_n$

Task:
find a satisfying assignment $a \in \mathrm{GF}[q]^n$
i.e. $p_1(a) = p_2(a) = \cdots = p_m(a) = 0$ holds

SysPolyEqs(q) as Hardness Assumption

Crypto-systems assuming the hardness of:

1. Enumerating all satisfying assignments

- Hidden Fields Equations (HFE) [Patarin'96,...]
- Unbalanced Oil and Vinegar signature schemes (UOV) [Kipnis-Patarin-Goubin'99,...]
- McEliece variants [Faugere-Otmani-Perret-Tillich'10,...]
- Polly cracker [Albrecht-Faugere-Farshim-Perret'11,...]

...

2. Finding one satisfying assignment

- QUAD [Berbain-Gilbert-Patarin'06,09,...]
- Matsumoto-Imai public key scheme [-'88,...]

...

SysPolyEqs(q) as Hardness Assumption

Strong Exponential Time Hypothesis (q^n is necessary) for SysPolyEqs(q) on degree 2 instances implies:

- The current best algorithm for the Listing Triangles problem is optimal [Björklund-Pagh-Vassilevska Williams-Zwick'14]
- Beating brute force for the GF(q)-weight k-clique problem is impossible [Vassilevska-Williams'09]

Previous Algorithms

- Groebner Basis: used in practice,
  double exponential time in the worst case

- $2^{n(1-\epsilon)}$ or polynomial time algorithms for SysPolyEqs(2) on degree 2 instances are known if instances satisfy some conditions e.g. [Yang-Chen'04,Bardet-Faugere-Salvy-Spaenlehauer'13,Miura-Hashimoto-Takagi'13,…]

- $q^{n/2}$ length ``proof" for the unsatisfiability of SysPolyEqs(q) on degree 2 instances [Woods'98]
  (i.e. co-nondeterministic algorithm for SAT)

Our Result 1
[randomized, search, bounded degree]

SysPolyEqs(q) on degree k instances can be solved in randomized time $q^{n(1-1/O(qk))}$

For $q = k = 2$, an important case for cryptography, we get the bound $\leq 2^{0.8765n}$

Input:
GF[q] polynomials $p_1, p_2, \ldots, p_m$
in formal variables $x_1, x_2, \ldots, x_n$

Task:
find a satisfying assignment $a \in \mathrm{GF}[q]^n$
i.e. $p_1(a) = p_2(a) = \cdots = p_m(a) = 0$ holds

Our Result 2

[deterministic, counting, bounded degree, prime field]

For a prime q, #SysPolyEqs(q) on degree k instances can be solved in deterministic time $q^{n(1-1/O(qk))}$

Input:

GF[q] polynomials $p_1, p_2, \ldots, p_m$

in formal variables $x_1, x_2, \ldots, x_n$

Task:

find a satisfying assignment $a \in \mathrm{GF}[q]^n$

i.e. $p_1(a) = p_2(a) = \cdots = p_m(a) = 0$ holds

Our Result 3

[deterministic, counting, unbounded degree, GF(2)]

For s =the total number of monomials, #SysPolyEqs(2) can be solved in deterministic time $2^{n(1-1/O(\log(s/n)))}$

Remark: exponentially faster than 2^n if $s = O(n)$

Input:

GF[q] polynomials $p_1, p_2, \ldots, p_m$

in formal variables $x_1, x_2, \ldots, x_n$

Task:

find a satisfying assignment $a \in \mathrm{GF}[q]^n$

i.e. $p_1(a) = p_2(a) = \cdots = p_m(a) = 0$ holds

Our Result 4
[deterministic, counting, unbounded degree, GF(2)]

GenSysPolyEqs(2)

Input:

- $\Sigma\Pi\Sigma$ circuits (sum of products of linear forms)
- $p_1, p_2, \ldots, p_m$ in formal variables $x_1, x_2, \ldots, x_n$
- e.g. $p_1 = (x_1 + x_2 + 1)(x_2 + x_3) + (x_1 + x_4)x_2 + 1$

Result:

For s =the total number of products of linear forms, #GenSysPolyEqs(2) can be solved in deterministic time $2^{n(1-1/O(\log(s/n)))}$

Remark: exponentially faster than 2^n if $s = O(n)$

Remark

(k-)CNF SAT is a special case of SysPolyEqs(2)

(on degree k instances)

e.g.

$C_1 = (\neg x_1 \vee x_2 \vee x_3) \Rightarrow p_1 = x_1(1+x_2)(1+x_3)$
$C_2 = (x_1 \vee \neg x_3 \vee \neg x_4) \Rightarrow p_2 = (1+x_1)x_2x_2$
$C_3 = (x_2 \vee x_3 \vee x_4) \Rightarrow p_3 = (1+x_1)(1+x_2)(1+x_3)$

$C_1 = C_2 = C_3 = 1 \Leftrightarrow p_1 = p_2 = p_3 = 0$

Optimality of Our Results

1. SysPolyEqs(2) on degree k instances can be solved in time $2^{n(1-1/O(k))}$

1′. k-CNF SAT can be solved in time $2^{n(1-1/k)}$

[Paturi-Pudlak-Zane'97,...]

2. For s =the total number of products of linear forms, GenSysPolyEqs(2) can be solved in time $2^{n(1-1/O(\log(s/n)))}$

2′. For s =the number of clauses,

CNF SAT can be solved in time $2^{n(1-1/(2\log(s/n)))}$

[Schuler'05,Calabro-Impagliazzo-Paturi'06,...]

Our Techniques

Polynomial Method in Circuit Complexity

(originally used for proving circuit size lower bounds)

We use (extensions of)

1. fast evaluation algorithms for polynomials [Yates'37,...]

2. approximation of polynomials by low degree probabilistic polynomials [Razborov'87,Smolensky'87] and its derandomization [Chan-Williams'16]

3. Schuler's width reduction for CNF-SAT [Schuler'05,...]

Cf. Polynomial Method

$\mathrm{AC}^0[q]$-circuit: bounded depth, unbounded-fan-in
Boolean circuit with AND/OR/NOT/mod q gates

Circuits Lower Bounds by [Razborov'87,Smolensky'87]:

1. $\mathrm{AC}^0[q]$-circuit can be well approximated by a low-degree GF(q) polynomial
2. $\mathrm{majority}, \mathrm{mod}\ r$ cannot be well approximated by a low-degree GF(q) polynomial
3. 1+2 ⇒ $\mathrm{majority}, \mathrm{mod}\ r \notin \mathrm{AC}^0[q]$-circuit

Item 1 is useful in algorithm design

Algorithms via Polynomial Method

(In what follows, we focus on GF(2))

Our Tool 1

Lemma[Fast Evaluation [Yates'37,...]]

Let $p: \{0,1\}^n \to \{0,1\}$ be a GF(2)-polynomial represented as a sum of monomials, then, the truth table of p can be generated in time $\mathrm{poly}(n)2^n$

Note:

The number of monomials in p can be 2^n

If we evaluate $p(x)$ for each $x \in \{0,1\}^n$, then it takes $\mathrm{poly}(n)4^n$

Basic Idea

Input: degree k polynomials $p_1, p_2, \dots, p_m$

1. Define $P: \{0,1\}^n \rightarrow \{0,1\}$ as

   $P \coloneqq (p_1+1)(p_2+1)\cdots(p_m+1)$

- $p_1(x) = p_2(x) = \cdots = p_m(x) = 0 \Leftrightarrow P(x) = 1$
- P might contain $\approx 2^n$ monomials

when represented as a sum of monomials

2. Define $R: \{0,1\}^{n-n'} \rightarrow \{0,1\}$ for some $n' < n$ as

   $R(y) \coloneqq \prod_{a \in \{0,1\}^{n'}} (P(y,a) + 1)$

- $\exists x, P(x) = 1 \Leftrightarrow \exists y, R(y) = 0$
- Each $P(y,a)$ might contain $\approx 2^{n-n'}$ monomials

Basic Idea

Observation:

If we can write $R(y)$ as a sum of monomials in time $2^{n-n\prime}$, we can also solve the problem in time $\text{poly}(n)2^{n-n\prime}$ by the Fast Evaluation Lemma

Note: straitfoward expansion needs $2^{n-n\prime} \times 2^{n\prime} \approx 2^{n}$

1. Define $P: \{0,1\}^n \to \{0,1\}$ as
   $P \coloneqq (p_1+1)(p_2+1) \cdots (p_m+1)$
2. Define $R: \{0,1\}^{n-n'} \to \{0,1\}$ for some $n' < n$ as
   $R(y) \coloneqq \prod_{a \in \{0,1\}^{n\prime}} (P(y,a) + 1)$

- $\exists x,\ p_1(x) = p_2(x) = \cdots = p_m(x) = 0 \Leftrightarrow \exists y, R(y) = 0$
- Each $P(y,a)$ might contain $\approx 2^{n-n\prime}$ monomials

Our Tool 2

Definition:

For $s_1, \ldots, s_d \in \{0,1\}^n$,

define degree d polynomial $Q_{\{s_i\}}: \{0,1\}^n \to \{0,1\}$ as

$Q_{\{s_i\}}(x) \coloneqq \prod_{i=1}^{d}\left((s_i, x) + 1\right)$, where $(s_i, x) \coloneqq \sum_{j\in[n]}(s_i)_j x_j$

Intuition: $Q_{\{s_i\}} \approx \prod_{i\in[n]}(x_i + 1)$

Lemma[Probabilistic Polynomial [Razborov'87,Smolensky'87]]

Select random $s_1, \ldots, s_d$ uniformly and independently,

then, for every non-zero $x \in \{0,1\}^n$,

$\Pr[Q_{\{s_i\}}(x) = 0] = 1 - 2^{-d}$ (cf. $\Pr[Q_{\{s_i\}}(0) = 1] = 1$)

Our Algorithm for degree k

Input: degree-k polynomials $p_1, p_2, \ldots, p_m$

1. Define $P: \{0,1\}^n \to \{0,1\}$ as
   $P := (p_1+1)(p_2+1)\cdots(p_m+1)$
2. Define $R: \{0,1\}^{n-n'} \to \{0,1\}$ for some $n' < n$ as
   $R(y) := \prod_{a\in\{0,1\}^{n'}}(P(y,a)+1)$
3. Replace each product by a probabilistic polynomial and write R as a sum of monomials p
4. Construct the truth table T of p
   if T contains an entry with 0, the input has a solution
5. Repeat 3-4 and take the majority voting of T's

Analysis of Our Algorithm

Input: degree-k polynomials $p_1, p_2, \ldots, p_m$

1. Define $P: \{0,1\}^n \to \{0,1\}$ as
   $P := (p_1 + 1)(p_2 + 1) \cdots (p_m + 1)$

2. Define $R: \{0,1\}^{n-n'} \to \{0,1\}$ for some $n' < n$ as
   $R(y) := \prod_{a \in \{0,1\}^{n'}} (P(y,a) + 1)$

3. Replace each product by a probabilistic polynomial
   and write R as a sum of monomials p

Step 3 takes time $\mathrm{poly}(n) 2^{n-n'}$
if each product is replaced by a low degree polynomial

Our Result 1

SysPolyEqs(q) on degree k instances can be solved in randomized time $q^{n(1-1/O(qk))}$

For $q = k = 2$, an important case for cryptography, we get the bound $\leq 2^{0.8765n}$

Input:

GF[q] polynomials $p_1, p_2, \dots, p_m$
in formal variables $x_1, x_2, \dots, x_n$

Task:

find a satisfying assignment $a \in \mathrm{GF}[q]^n$
i.e. $p_1(a) = p_2(a) = \cdots = p_m(a) = 0$ holds

On Deterministic Algorithms

1. Derandomization of probabilistic polynomials due to Razoborov-Smolensky [Chan-Williams'16]

- small biased space [Naor-Naor'90,...]
- modulus amplifying polynomial [Toda'89,Yao'90,Beigel-Tarui'91]

2. Fast evaluation algorithms for non-multilinear integer polynomials

- fast rectangular matrix multiplication [Coppersmith'82,...,LeGall 12]

Our Algorithm for unbounded degree

Input: polynomials $p_1, p_2, \dots, p_m$

1. [Degree Reduction]
   generate exponentially many instances
   of SysPolyEqs(2) such that

   (1) original input has a solution if and only if
   at least one of generated instances has a solution

   (2) generated instances have degree at most k

2. Apply the algorithm for degree k

Our Algorithm for unbounded degree

Degree Reduction:

while there is a monomial of degree $> k$

e.g. $p_1 = x_1 \dots x_k x_{k+1} \dots x_n + \cdots$

generate two instances as

I-1: $x_1 \dots x_k = 1$, i.e., $x_1 = \cdots = x_k = 1$

I-2: $x_1 \dots x_k = 0$ (added as a polynomial equation)

Note: Degree reduction can be generalized to handle $\Sigma\Pi\Sigma$ circuits (sum of products of linear forms) by ``Simplification Rules" based on ``change of basis" in Linear Algebra

Conclusion

Our Results

1. SysPolyEqs(q) on degree k instances can be solved in randomized time $q^{n(1-1/O(qk))}$

2. For $q = k = 2$, an important case for cryptography, we get the bound $\leq 2^{0.8765n}$

3. For a prime q, #SysPolyEqs(q) on degree k instances can be solved in deterministic time $q^{n(1-1/O(qk))}$

4. For $s =$the total number of products of linear forms, #GenSysPolyEqs(2) can be solved in deterministic time $2^{n(1-1/O(\log(s/n)))}$

Optimality: Improvement requires that for (k-)CNF SAT

Future Directions

- Similar running time in polynomial space
- Degree Reduction for PolySysEqs(q), $q \neq 2$
- Beating Brute Force for other problems using the Polynomial Method
- Develop/Apply Fast Evaluation Algorithms for more expressive classes than polynomials

Thank you for your attention!