A brief history of polynomial identity testing

Manindra Agrawal*

Department of Computer Science and Engineering, Indian Institute of Technology Kanpur, Kanpur 208 016, India

Polynomial identity testing is the problem of deciding if a given (multivariate) polynomial is identically zero. Over the past decade, connections of this problem with a fundamental issue of complexity theory have been discovered and so the problem has attracted a lot of attention. In this article, we provide a brief history of the problem and its current status.

Keywords: Arithmetic circuits, black-box algorithm, complexity theory, polynomial identity testing.

Introduction

POLYNOMIAL Identity Testing (PIT) is the problem of checking if a polynomial of n variables over a field F is identically zero, i.e. if all its terms cancel each other out. The problem is simple to solve efficiently if the polynomial is given in the usual sum-of-products form

$$P(x_1, x_2, ..., x_n) = \sum_{0 \le i_1, i_2, ... i_n \le d} c_{i_1, i_2, ..., i_n} x_1^{i_1} x_2^{i_2} ... x_n^{i_n},$$

simply check whether all coefficients $c_{i_1, i_2, ..., i_n} \in F$ are zero. The problem becomes non-trivial when the polynomial is given in an implicit form, e.g. as determinant of a matrix with polynomial entries. One can compute the determinant polynomial to express it in the above form. However, the size of the polynomial can grow exponentially in this process and so this algorithm is exponential overtime. Finding an efficient algorithm for the problem has over the years become one of the most important challenges in complexity theory with fundamental implications.

A general representation of polynomials is via arithmetic circuits: these define a sequence of addition and multiplication operations starting from variables and ending in the desired polynomial. For example

$$P(u, v, x, y) = (ux + vy)^2 + (vx - uy)^2 - (u^2 + v^2) \cdot (x^2 + y^2).$$

Figure 1 represents an arithmetic circuit computing polynomial P on four variables u, v, x and y. The operations are inside circles and take as input polynomials on arrows coming into the circles and output the resulting polynomial on arrows going out of the circles. If a constant c is present on an arrow carrying polynomial Q into an addition operation, the polynomial is replaced by cQ, and if the arrow is going into a multiplication operation, the polynomial is replaced by Q^c.

Arithmetic circuits provide a natural and succinct way of representing polynomials. Essentially, an arithmetic circuit represents a quick method to compute a polynomial on any given point (by performing operations in sequence). It is known that any polynomial that can be expressed as determinant of a matrix, with entries being constant or variables, can be represented by an arithmetic circuit of size polynomial in the matrix size. Here, the size of an arithmetic circuit is defined as the number of operations in the circuit. So, for example, the size of the above circuit is 16. Another important parameter associated with an arithmetic circuit is its depth. The depth of an arithmetic circuit is the length of the longest chain of arrows from an input variable to the output polynomial. In the above example circuit, the depth is 4.

There is an important restriction of the problem: low degree PIT (LPIT). This is the problem when there exists a polynomial bound $r(\cdot)$ such that the degree of the polynomial represented by a given circuit is bounded by $r(s)$, where s is the size of the circuit. Most of the commonly encountered instances of PIT are of this form.

History: 20th century

For the purpose of describing algorithms given below for solving the problem, we assume that a polynomial P of n variables over a field F is given in the form of an arithmetic circuit C of size s and depth d. Further, P is non-zero (if P is zero, all the tests below trivially work).

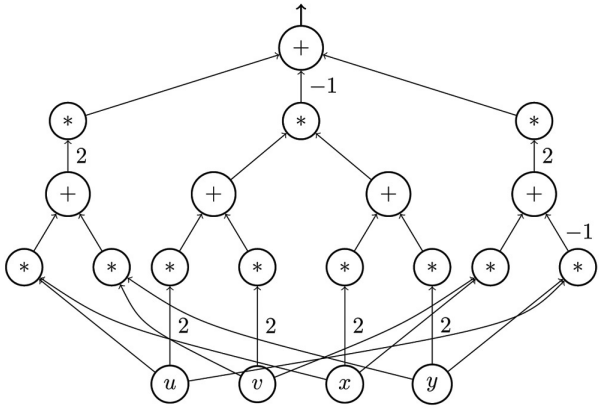

Figure 1. Example of an arithmetic circuit.

*e-mail: manindra@iitk.ac.in

The earliest work on this problem was reported in the late 1970s in two independent papers by Schwartz[1] and Zippel[2]. Both of them showed a randomized polynomial time algorithm for solving the problem. Their algorithms were based on the following observation.

Lemma 1. *Suppose the degree of each variable of P is bounded by D. Then, for any set of values $S \subseteq F$, the fraction of points in S^n on which P is zero is bounded by $D/|S|$.*

Therefore, by choosing S to be any set of $2D$ values (which is bounded by exponential in the depth of the circuit), and randomly selecting a point in S^n, the probability that P will be zero on this point is at most 1/2.

The next significant development took place nearly 20 years after this. Chen and Kao[3] designed a randomized polynomial time algorithm for the LPIT problem over $\mathbb{Q}$ that used a very different idea: instead of picking a random point from a set, they defined one irrational point $\bar{\alpha}$ such that $P(\bar{\alpha}) \neq 0$. Then they showed that P is non-zero on most of a set of D^n rational points in the neighbourhood of α. Further, the number of points on which P is non-zero increases with closeness of these points to α. Soon after, Lewin and Vadhan[4] extended this idea to work over finite fields, and Agrawal and Biswas[5] generalized the idea using univariate polynomials to obtain a similar test for the PIT problem.

Theorem 1. *Let $Q_{t,\bar{b}}(x) = x^t + \sum_{i=0}^{s-1} b[i]x^i$ with $\bar{b}$ being a bit-vector of length s. Then for $t \geq s$, for at most $2^s/t$ bit-vectors $\bar{b}$,*

$$P(x, x^{2s}, x^{2^{2s}}, \ldots, x^{2^{(n-1)s}}) = 0 \pmod{Q_{t,\bar{b}}^r - 1},$$

for all $r \leq s^2 t$.

The above theorem suggests a test in which the error probability can be decreased by increasing t without increasing the amount of randomness required.

History: 21st century

Two results in 2002, within months of each other, catapulted PIT to the centre stage of complexity theory. Kabanets and Impagliazzo[6] showed that a deterministic polynomial-time algorithm for LPIT would yield a super-polynomial lower bound on an explicit polynomial family.

Theorem 2. *If there is a deterministic polynomial-time algorithm for LPIT, then there is an explicit family of polynomials in NEXP (non-deterministic exponential time) that requires superpolynomial size arithmetic circuits.*

An explicit family of polynomials is simply a polynomial family that is computable within a certain time bound. For example, the polynomial family $\{x_1 + x_2 + \cdots + x_n\}_{n \geq 1}$ is explicit since, given n, the corresponding polynomial in the family can be computed in time $O(n)$. Finding an explicit family of polynomials that requires superpolynomial size circuits to compute is the central problem of arithmetic complexity theory. Although we know that most of the polynomial families require superpolynomial size circuits to compute (by a simple counting argument), we do not yet have an example of an explicit family computable in exponential time (EXP) that requires superpolynomial size. The above theorem suggests a path to achieve this. Agrawal *et al.*[7] showed how to obtain such a deterministic algorithm for a special class of polynomials[5] that characterize prime numbers.

Theorem 3. *Number n is prime if $P_n(x) = (1 + x)^n - x^n - 1 = 0 \pmod{n}$. Further, for any composite n, $P_n(x) \neq 0 \pmod{n, Q_n(x)}$, where $Q_n(x)$ is an explicit polynomial of degree $O(\log^{14} n)$.*

This resulted in the first deterministic polynomial time algorithm for testing if a number is prime, and also provided reasons to believe that the approach suggested by Kabanets and Impagliazzo[6] is tractable. It led to more attention to the PIT problem, and over the last decade much progress has been made towards solving it.

The next important advance was an observation by Agrawal[8] that deterministic polynomial-time black-box algorithms for LPIT result in stronger lower bound. A black-box algorithm for PIT is an algorithm that, given circuit C computing polynomial P, produces several points, whose values depend only on the size of C, such that P is not zero on at least one of these points. The name 'black-box' comes from the fact that such an algorithm does not need to know the structure of the circuit C, and so C can be hidden inside a 'black-box' and yet the algorithm will work correctly. Black-box algorithms exist for even PIT: all the randomized algorithms described above are black-box algorithms for PIT or LPIT. The observation of Agrawal[8] was actually already made in 1980 by Heintz and Schnorr[9].

Theorem 4. *If there is a deterministic polynomial-time black-box algorithm for LPIT, then there is an explicit family of polynomials in EXP (deterministic exponential time) that requires exponential size arithmetic circuits.*

A partial converse of this theorem also holds[6,8].

Theorem 5. *If there is an explicit family of polynomials in EXP that requires exponential size arithmetic circuits, then there is a deterministic, $s^{O(\log s)}$ time, black-box algorithm for LPIT.*

In parallel, researchers were studying the types of polynomial families for which deterministic polynomial-time

algorithms can be designed. All such attempts (e.g. Kayal and Saxena[10]) could only work for special families of depth-three polynomials. On the other hand, it was known[11,12] that any LPIT circuit of size s can be converted to a circuit of similar size and depth $O(\log s)$. Hence there was seemingly a large gap between what we could show and what we needed to. However, this gap nearly vanished soon after[13].

Theorem 6. *If there is a deterministic polynomial-time black-box algorithm for LPIT restricted to depth-four circuits, then there is an explicit family of polynomials in EXP that requires exponential size arithmetic circuits.*

The above result showed that in order to derive exponential size lower bounds, it is enough to design a deterministic polynomial-time black-box algorithm for depth-four circuit families. This gave further impetus to the investigations into LPIT for small depth circuit families. A number of results followed, for special families of depth-three and depth-four polynomials. The strongest of these show as follows[14,15].

Theorem 7. *There exists a deterministic, $s^{O(k)}$ time, black-box algorithm for LPIT restricted to depth-three circuits whose top gate has at most k inputs.*

Thus, to get lower bounds, one needs to extend the above result first to arbitrary depth-three circuits and then to depth-four circuits. Very recently, the second step has been shown to be superfluous[16].

Theorem 8. *If there is a deterministic polynomial-time black-box algorithm for LPIT restricted to depth-three circuits and over fields of large characteristic, then there is an explicit family of polynomials in EXP that requires exponential size arithmetic circuits.*

This brings the target tantalizingly close: one only needs to get a time bound of $(sk)^{O(1)}$ instead of $s^{O(k)}$ in Theorem 7.

1. Schwartz, J. T., Fast probabilistic algorithms for verification of polynomial identities. *J. ACM*, 1980, **27**(4), 701–717.
2. Zippel, R. E., Probabilistic algorithms for sparse polynomials. In EUROSCAM'79, Springer LNCS 72, 1979, pp. 216–226.
3. Zhi-Zhong Chen and Ming-Yang Kao, Reducing randomness via irrational numbers. In Proceedings of Annual ACM Symposium on the Theory of Computing, 1997, pp. 200–209.
4. Lewin, D. and Vadhan, S., Checking polynomial identities over any field: towards a derandomization? In Proceedings of Annual ACM Symposium on the Theory of Computing, 1998, pp. 428–437.
5. Agrawal, M. and Biswas, S., Primality and identity testing via chinese remaindering. *J. ACM*, 2003, **50**(4), 429–443.
6. Kabanets, V. and Impagliazzo, R., Derandomizing polyonmial identity tests means proving circuit lower bounds. *Comput. Complexity*, 2004, **13**, 1–46.
7. Agrawal, M., Kayal, N. and Saxena, N., *PRIMES is in P. Ann. Math.*, 2004, **160**(2), 781–793.
8. Agrawal, M., Proving lower bounds via pesudo-random generators. In Proceedings of the FST&TCS, 2005, pp. 96–105.
9. Heintz, J. and Schnorr, C. P., Testing polynomials which are easy to compute. In Proceedings of Annual ACM Symposium on the Theory of Computing, 1980, pp. 262–268.
10. Kayal, N. and Saxena, N., Polynomial identity testing for depth 3 circuits. *Comput. Complexity*, 2007, **16**(2), 115–138.
11. Valiant, L., Skyum, S., Berkowitz, S. and Rackoff, C., Fast parallel computation of polynomials using few processors. *SIAM J. Comput.*, 1983, **12**, 641–644.
12. Allender, E., Jiao, J., Mahajan, M. and Vinay, V., Non-commutative arithmetic circuits: depth reduction and size lower bounds. *Theor. Comput. Sci.*, 1998, **209**, 47–86.
13. Agrawal, M. and Vinay, V., Arithmetic circuits: a chasm at depth four. In Proceedings of Annual IEEE Symposium on Foundations of Computer Science, 2008, pp. 67–75.
14. Saxena, N. and Seshadri, C., From Sylvester–Gallai configurations to rank bounds: improved black-box identity test for depth 3 circuits. *J. ACM*, 2013, **60**(5).
15. Agrawal, M., Saha, C., Saptharishi, R. and Saxena, N., Jacobian hits circuits: hitting sets, lower bounds for depth-d occur-k formulas and depth-3 transcendence degree k circuits. In Proceedings of Annual ACM Symposium on the Theory of Computing, 2012, pp. 599–614.
16. Gupta, A., Kamath, P., Kayal, N. and Saptharishi, R., Arithmetic circuits: a chasm at depth three. In Proceedings of Annual IEEE Symposium on Foundations of Computer Science, 2013, pp. 578–587.

ACKNOWLEDGEMENT. This research was supported by the J. C. Bose Fellowship FLW/DST/CS/20060225.