DOI: 10.1145/3631338

BY PROF.DR. NITIN SAXENA

How Easy Is It to Describe Hard Polynomials?

We study real-life problems via mathematical models, which usually tend to be nonlinear systems, or equations. A standard trick is to "linearize" this nonlinear system. The newly obtained system then becomes a subject of linear algebra; at which point we get the happy feeling that we have solved the original nonlinear system. A key object in this path to happiness is the determinant. Practitioners in science and engineering use the determinant to solve a linear system, which is the bedrock of algorithms in optimization, data analysis, graphics, game theory, economics, and business.

The following paper studies a related polynomial that is easier to describe: Take d square matrices, each of order n, multiply them, and consider the top-left entry of the product matrix; this defines the iterated matrix multiplication (IMM) polynomial in dn^2 variables. How compactly can the polynomial IMM be fabricated on a machine? A positive answer to this question would give us practical algorithms, while a negative answer would mean we have identified a hard polynomial that is easy to describe.

If we fabricate term by term, a naïve count says there are n^d terms in IMM. This is exponential in d and becomes impossibly large as the input parameter d grows, say, beyond 100. We want to optimize this situation or prove that no further optimization is possible to fabricate IMM on machine models. The computer science area that studies such questions is *algebraic complexity*, and it is a rapidly growing subarea.

Algebraic complexity develops ideas that are useful in two ways: Upper bound—There are tools to solve problems like computing the determinant and IMM very fast, finding the root of a polynomial system, finding a polynomial factor, and testing whether two polynomials are the same. Lower bound—There are methods to prove that certain natural looking polynomials are hard to fabricate and compute.

Why should a practitioner care about hard polynomials? Shouldn't our focus, as computer scientists, be on only solving problems; say, by adding interesting simplifying assumptions? Though that may be our ideal job, there are many problems out there that seem to possess an inherent intractability so that even practical assumptions do not help to solve them in any meaningful way. Surprisingly, such problems may be a boon to other areas; for example, cryptography turns hard problems into practical protocols secure to adversarial attacks. This is what secures the Internet. Thus, it motivates us to find not only an upper bound, whenever possible, but also a lower bound proof.

A famous example is the problem of counting the number of perfect matchings in a graph. For instance, you may want to assign tasks to servers in a stable way, given the preference list of each party involved. How many such assignments are possible? This counting problem lives at the heart of algebraic complexity theory and exactly defines a polynomial called permanent, per(M), given a square matrix M. So, now we have brought on the stage two polynomials: IMM and per(M). It is widely conjectured that per(M) is the more difficult one among the two; proving its hardness is the central open question in algebraic complexity (called the VP≠VNP question).

In contrast, it is not difficult to see that IMM is friendlier, in the sense that a clever circuit can express IMM in a compact way. However, it was an open question to understand the depth of this circuit. In the spirit of parallel processing, the circuit depth signifies the time taken to compute the polynomial; while the size refers to the space or the number of arithmetic processors required to fabricate the polynomial. The paper proves that if we restrict to constant-depth then IMM requires a circuit of very large size. Roughly put, constant-time matrix multiplication requires exponentially many arithmetic processors.

Not only is this lower-bound statement prized, more interesting is the proof method that it develops. It has two important lessons. First, the setting where d (= number of matrices) is significantly smaller than n (= order of each matrix) is amenable to better structural transformations. This allows us to make the circuit multiplication gates very well-behaved (namely, set-multilinear). Second, there is a low-rank matrix associated with these multiplication gates (namely, the partial-derivative matrix). This shows us that IMM has a low-rank matrix, which yields a contradiction, unless the circuit size is superpolynomial.

In the past decades, algebraic complexity has taken big strides in the development of techniques. This paper achieves a landmark in the larger quest of understanding hardness, identity testing, and reconstruction. It encourages us to try newer circuit transformations that linearize a circuit, just enough, to apply linear algebra.

Nitin Saxena is the N. Rama Rao Chair Professor in the Department of Computer Science and Engineering at the Indian Institute of Technology Kanpur, India. Follow his work at https://bit.ly/40w8liJ.