Symbolic Computation and Complexity Theory Transcript of My Talk

Erich L. Kaltofen

Key words: Computational complexity, exponential-time algorithms, practicality

1 The Setting

I gave talks at the conference *Alan Turing's Heritage: Logic, Computation & Complexity* in Lyon, France on July 3, 2012, at the Pierre and Marie Curie University (UPMC) Paris 6, France on July 17, 2012, and at the Tenth Asian Symposium on Computer Mathematics (ASCM) in Beijing, China on October 26, 2012 on the complexity theoretic hardness of many problems that the discipline of symbolic computation tackles. Here is a brief transcript of part of those talks.

1.1 NP-completeness and Beyond

A fundamental problem of symbolic computation, that of solving systems of polynomial equations, is easily shown to be NP-hard: $x \vee \neg y \quad \equiv \quad (1-x)y = 0, x(x-1) = 0, y(y-1) = 0$, which shows how to encode a clause in a satisfiability problem as polynomial equations.

Real geometry, when the solutions of the polynomial systems are restricted to the real numbers, is by Tarski's algorithm decidable. However, [Fischer and

Department of Mathematics, North Carolina State University, Raleigh, North Carolina 27695-8205, USA kaltofen@math.ncsu.edu; http://www.kaltofen.us

* This material is based on work supported in part by the National Science Foundation under Grants CCF-1115772.

Rabin 1974] have shown that the problem requires exponential space, $2^{\Omega(n)}$, where n is the number of variables in the polynomials. Furthermore, [Mayr and Meyer 1982] have extended the result to Polynomial Ideal Membership over the rationals, that is, they show $2^{\Omega(n)}$-space hardness.

Finally, [Fröhlich and Shepherdson 1955/56] show that there are fields K in which the five arithmetic operations, namely addition, negation, multiplication, division, and equality testing are computable but where factorization in $\mathsf{K}[x]$ is undecidable ("unentscheidbar"). The proof is based on the infinite tower of extensions by squareroots of prime integers and the fact that $\sqrt{2} \notin \mathbb{Q}(\sqrt{3}, \sqrt{5}, \sqrt{7}, \sqrt{11}, \ldots)$, for instance.

These are indeed formidable computational complexity theoretic barriers to the discipline of symbolic computation.

1.2 Early Symbolic Computation Algorithms

Buchberger's famous 1965 Gröbner basis algorithm tackles exactly those hard problems: it decides ideal membership and computes solutions to polynomial systems. Berlekamp's and Zassenhaus's 1968 polynomial factorization algorithms work for coefficients in finite fields and the rational numbers. Collins's 1974 cylindrical algebraic decomposition algorithm performs Tarski's quantifier elimination.

The pursuit of symbolic computation algorithms that solve those computational hard problems in the early 1980s was ridiculed by some theorists as hopeless.

2 Cook's Thesis

In his plenary talk at the ICM in Kyoto [Cook 1990], three function classes are introduced:
polytime, the functions computable by polynomial-time algorithms,
NAT, functions arising from natural computational problems,
PracSolv, functions computable on an actual computer on **all inputs** of 10,000 bits or less.

Stephen Cook then formalizes his thesis:

Thesis: $\mathit{PracSolv} \cap \mathit{NAT} = \mathit{polytime} \cap \mathit{NAT}$.

The notion that *polytime* captures the domain of efficiently computable functions is ingrained in theoretical computer science. Many reductions in modern theoretical cryptography make use of the device. Stephen Cook refines the notion to natural functions. As an unnatural function he gives the example of $2^{\lceil \log n \rceil^{1000}} \in \mathit{polytime}$, that with the high exponent of 1000, and which he excludes from NAT.

As evidence in 1990 there were the polynomial-time algorithms for linear programming and for polynomial factorization over the rational numbers. Polynomial identity testing via the Zippel-Schwartz lemma was and is random polynomial time, and at that time it was hypothesized that with randomization in polynomial-time algorithms one could not reach beyond the class *polytime*. Today, by results by Impagliazio and Kabanets there is more doubt. Indeed, already in 1990 polynomial factorization, even for polynomials in many variables, was know to be in randomized *polytime* [Kaltofen and Trager 1990]. Supersparse polynomial factorization algorithms would follow 15 years later [Kaltofen and Koiran 2006].

Richard D. Jenks had expressed some doubt in the thesis at that time, telling me as PhD student: *"You prove that problems are hard and I write computer programs that solve them."* In the following I will attempt to challenge Cook's Thesis.

The Thesis can fail in two directions. There may exist an $f \in \mathit{PracSolv} \cap \mathit{NAT}$ but $f \notin \mathit{polytime}$. Many programs in symbolic computation can produce outputs to problems that are, in the worst case, hard.

1. Proofs that a positive semidefinite polynomial is not a sum-of-squares: 462-dimensional linear matrix inequalities (LMI) with 7546 variables [Guo et al. 2012]. Semidefinite programming constitutes a far-reaching generalization to linear programming with a limited non-linearity in its control parameters: the solution must remain a definite matrix.
2. Large Gröbner basis problems, e.g., compact McEliece crypto system: 115 variables, 193584 equations, degrees $= 2, 3, \ldots, 256$ [Faugère et al. 2010].
3. Proofs that certain non-linear polynomial equations problems (LMIs) do not have a rational solution, while they have a real solution [Guo et al. 2013]. I have substituted this diophantine problem to my list for this paper.

The above examples do not violate Cook's Thesis. Clearly, a super-polynomial-time algorithm can have polynomial-time running time on a subset of inputs. Many algorithms in symbolic computation, such as Buchberger's algorithm and its modern variant *FGb*, have an unpredictable running time. Cook's *NAT* is the class of "natural" functions, not "natural" inputs. It is important for algorithmic infrastructure to know the worst-case behavior of an algorithm: Google returns a list of ranked pages for all queries. Nonetheless, it is the hallmark of the symbolic computation discipline to have greatly enlarged the domain of natural and solvable inputs to hard problems.

The Thesis can also fail in the opposite: We may have an $f \in \mathit{polytime} \cap \mathit{NAT}$ but $f \notin \mathit{PracSolv}$. It is actually not so easy to find a natural problem in symbolic computation that is in *polytime* but whose worst-case complexity is super-quadratic in its input size. I offer three examples:

1. The characteristic polynomial of a sparse matrix $\in \mathbb{Z}_p^{n\times n}$ with $O(n)$ non-zero entries is notoriously difficult to compute with $O(n)$ auxiliary space

in the worst case. The best algorithm is of $n^{2+1/2+o(1)}$-time, $O(n)$-space [Villard 2000]. For $n = 10^6$ this is, ignoring the implied $n^{o(1)}$, $\geq 10^9\times$ input size. We restrict to $O(n)$ space because by using quadratic space one has $O(n^{2.38})$ time with fast matrix multiplication, although that solution is quite impractical.

2. The Sylvester resultant in x of $f(x, y), g(x, y)$ can be computed by the half-GCD algorithm in $\max\{\deg_x(f), \deg_x(g)\}^{2+o(1)} \times \max\{\deg_y(f), \deg_y(g)\}^{1+o(1)}$ scalar operations.
3. Lattice basis reduction is polynomial-time, but the dependency on the dimension may be super-quadratic. I have substituted this diophantine problem to my list for this paper.

In sparse/structured linear algebra, $O(n \log(n))$ vs. $O(n^2)$ running time makes all the difference, for example in discrete Fourier transform algorithms. Polynomial factorization is again a fore-runner: polynomials modulo 2 can be factored in subquadratic time since [Kaltofen and Shoup 1998].

Shaoshi Chen has told me while he was at NCSU that some of the algorithms for symbolic summation have worst case performance beyond quadratic. Those and the above are all candidates for polynomial-time problems that are not practically solvable for large inputs, although one may require much more than Cook's original 10,000 bits for the inputs. It is my conclusion that the Thesis fails on that side: $\mathit{PracSolv} \cap \mathit{NAT} \subsetneq \mathit{polytime} \cap \mathit{NAT}$.

Stephen Cook at the Turing Centennial celebration in San Francisco in June 2012 has suggested to me to consider in place of *polytime* the class of *logarithmic space* as the practical one (in my talk I stated it as *poly-logarithmic space*).

3 Faugère's Question

After my talk in Paris, Jean-Charles Faugère asked me the following question: does it make sense to study **and implement** algorithms that have exponential running time? My answer was "no," with some disapproval from the audience. I clarified that one may study algorithms that are worst-case exponential, but that run polynomial-time on the inputs studied. Executing an exponential-time algorithm, say a combinatorial search, constitutes a single computation, not providing an algorithm for general use.

I should add Ludovic Perret's comment to me at NCSU in October 2012: One studies exponential algorithms to know their run times, for example when choosing size of a key in a crypto scheme.

References

Stephen. A. Cook. Computational complexity of higher type functions. In *Proc. ICM Kyoto Japan*, ICM Series, pages 55–69, 1990.

Jean-Charles Faugère, Ayoub Otmani, Ludovic Perret, and Jean-Pierre Tillich. Algebraic cryptanalysis of McEliece variants with compact keys. In *Proc. Eurocrypt 2010*, volume 6110 of *Lect. Notes Comput. Sci.*, pages 279–298, Heidelberg, Germany, 2010. Springer Verlag.

M. J. Fischer and M. O. Rabin. Super-exponential complexity of presburger arithmetic. In R. M. Karp, editor, *Complexity of Computation*, pages 27–41. Amer. Math. Soc., 1974.

A. Fröhlich and J. C. Shepherdson. Effective procedures in field theory. *Phil. Trans. Roy. Soc., Ser. A*, 248:407–432, 1955/56.

Feng Guo, Erich L. Kaltofen, and Lihong Zhi. Certificates of impossibility of Hilbert-Artin representations of a given degree for definite polynomials and functions. In Joris van der Hoeven and Mark van Hoeij, editors, *ISSAC 2012 Proc. 37th Internat. Symp. Symbolic Algebraic Comput.*, pages 195–202, New York, N. Y., July 2012. Association for Computing Machinery. ISBN 978-1-4503-1269. URL: http://www.math.ncsu.edu/~kaltofen/bibliography/12/GKZ12.pdf; URL: http://arxiv.org/abs/1203.0253.

Q. Guo, M. Safey El Din, and L. Zhi. Computing rational solutions of linear matrix inequalities. In Manuel Kauers, editor, *ISSAC 2013 Proc. 38th Internat. Symp. Symbolic Algebraic Comput.*, New York, N. Y., 2013. Association for Computing Machinery.

E. Kaltofen and V. Shoup. Subquadratic-time factoring of polynomials over finite fields. *Math. Comput.*, 67(223):1179–1197, July 1998. URL: http://www.math.ncsu.edu/~kaltofen/bibliography/98/KaSh98.pdf.

E. Kaltofen and B. Trager. Computing with polynomials given by black boxes for their evaluations: Greatest common divisors, factorization, separation of numerators and denominators. *J. Symbolic Comput.*, 9(3):301–320, 1990. URL: http://www.math.ncsu.edu/~kaltofen/bibliography/90/KaTr90.pdf.

Erich Kaltofen and Pascal Koiran. Finding small degree factors of multivariate supersparse (lacunary) polynomials over algebraic number fields. In Jean-Guillaume Dumas, editor, *ISSAC MMVI Proc. 2006 Internat. Symp. Symbolic Algebraic Comput.*, pages 162–168, New York, N. Y., 2006. ACM Press. ISBN 1-59593-276-3. URL: http://www.math.ncsu.edu/~kaltofen/bibliography/06/KaKoi06.pdf.

E. W. Mayr and A. R. Meyer. The complexity of the word problem for commutative semigroups and polynomial ideals. *Advances Math.*, 46:305–329, 1982.

Gilles Villard. Computing the Frobenius normal form of a sparse matrix. In *Proc. Third International Workshop on Computer Algebra in Scientific Computing*, pages 395–407, Heidelberg, Germany, 2000. Springer Verlag.