The Rôle of Benchmarking in Symbolic Computation (Position Paper)

James Davenport
Thanks to Alessandro Cimatti for the SMT graphs

University of Bath

21 September 2018

Structure

There is little doubt that, in the minds of most symbolic computation researchers, the ideal paper consists of a problem statement, a new algorithm, a complexity analysis and preferably a few validating examples. There are many such great papers. This paradigm has served computer algebra well for many years, and indeed continues to do so where it is applicable. However, it is much less applicable to sparse problems, where there are many NP-hardness results, or to many problems coming from algebraic geometry, where the worst-case complexity seems to be rare.
We argue that, in these cases, the field should take a leaf out of the practices of the SAT-solving community, and adopt *systematic* benchmarking, and benchmarking contests, as a way measuring (and stimulating) progress. This would involve a change of culture.

Complexity Theory

Symbolic computation was an early beneficiary [Knu69] of rigorous complexity theory. This led to the paradigm that the ideal paper consists of a problem statement, a new algorithm, a complexity analysis and preferably a few validating examples. There are many such great papers [Bro71, Col67, LLL82].
This works fairly well for dense polynomials, where complexity is a function of degree (and coefficient length), but essentially all algebra systems implement sparse polynomials, at least "sparse in coefficient".

Multiplication

Open Problem ([Roc18, Problem 1])

Develop an algorithm to multiply two sparse polynomials $f, g \in R[x]$ using $\widetilde{O}(t \log D)$ ring and bit operations, where t is the number of terms in f, g and fg, and D is an upper bound on their degree.

Provided R is $\mathbf{Z}$ or similar, we can almost solve this: $\widetilde{O}\left(\hat{t} \log D\right)$ where $\hat{t}$ is the number of terms in f, g *and* a generic $\hat{f}\hat{g}$, where $\hat{f}$ has the same exponents as f etc. In other words, a "no cancellation" version of fg.
I think this is reasonable: if t is much less than $\hat{t}$ it means there's been a great deal of cancellation *in this case*..

Division 1

The example of $\frac{x^n-1}{x-1} = x^{n-1} + \cdots x + 1$ shows that we need to consider the number of terms in the output, as well as in the input.

Open Problem ([Roc18, Problem 2])

Given two sparse polynomials $f, g \in R[x]$, develop an algorithm to compute the quotient and remainder $q, r \in R[x]$ such that $f = qg + r$, using $\widetilde{O}(t \log D)$ ring and bit operations, where t is the number of terms in f, g and q and r, and $\deg f < D$.

Not "nearly in reach" when g is sparse — when g is dense we compute powers of x modulo g.
Even the decision problem "does g divide f exactly" is unknown, and this is key to verifying some modular algorithms.

Division 2

Open Problem ([DC10, Challenge 3])

Either

- *find a class of problems for which the problem "does g divide f?" is NP-complete; or*
- *find an algorithm for the divisibility of polynomials which is polynomial-time.*

A weaker version of this might be to exclude the (possibly scaled) cyclotomics: again unknown.

Greatest Common Divisors 1

Again it is necessary to consider output size, as the neat example of [Sch03] shows:

$$\gcd(x^{pq}-1, x^{p+q}-x^p-x^q+1) = x^{p+q-1}-x^{p+q-2} \pm \cdots - 1.$$

Most of the classic results in this are are due to Plaisted [Pla77, Pla78, Pla84], as in the following result.

Theorem ([Pla78])

It is NP-hard to determine whether two sparse polynomials (in the standard encoding) have a non-trivial common divisor.

Of course, there aren't that many ways of proving things NP-hard.

Greatest Common Divisors 2

> *The basic device of the proofs is to encode the NP-complete problem of 3-satisfiability so that a formula W in n Boolean variables goes to a sparse polynomial $p_M(W)$ which vanishes exactly at certain Mth roots of unity corresponding to the satisfiable assignments to the formula W, where M is the product of the first n primes. [MR 85j:68043]*

But, of course, not all n-SAT problems are hard: just enough of them!

Greatest Common Divisors 3: non cyclotomics

All known hard cases seem to involve(versions of) $x^n - 1$.

Open Problem ([DC10, Challenge 2])

Either

- *find a class of problems for which the g.c.d. problem is still NP-complete even when cyclotomic factors are explicitly encoded (see paper APpendix A); or*
- *find an algorithm for the g.c.d. of polynomials with no cyclotomic factors, which is polynomial-time in the standard encoding.*

As this is undecided, the state of the art seems to be that even the decision problem (output size one bit) for greatest common divisors can be NP-hard on some (probably rare) problems.

Polynomial Factorization: Square-free

Practically all known polynomial factorization algorithms begin by doing a square-free decomposition, and this is also hard in theory.

Theorem ([KS99])

Over $\mathbf{Z}$ *and in the standard sparse encoding, the two problems*

1. *deciding if a polynomial is square-free*
2. *deciding if two polynomials have a non-trivial g.c.d.*

are equivalent under randomized polynomial-time reduction.

Hence, in the light of Plaisted's Theorem, determining square-freeness is hard, at least when polynomials with cyclotomic factors are involved.
In practice, I think we'll always do a square-free.
Please prove me wrong!

Polynomial Factorization: Dense

Even in the dense case, very little is known about the worst-case complexity of polynomial factorization (beyond the fact that it's in P [LLL82]), due to the existence of Swinnerton-Dyer polynomials (those that factor compatibly modulo every prime, but are irreducible). Since almost all polynomials are irreducible in the sense that $\forall d > 0$

$$\lim_{H\to\infty} \frac{|\{\text{such polynomials that factor}\}|}{|\{\text{polynomials of degree } d, \text{ coefficients} \le H\}|} = 0, \qquad (1)$$

typical-case complexity isn't helpful.
Hence polynomial factorization papers nearly always rely on a set of examples to demonstrate their superiority (e.g. [Wan78] drawing on [Cla76]). Hardware progress (as well as some algorithmic improvements) have made this particular set of problems trivial, and there doesn't seem to be an agreed corpus of hard problems.

Polynomial Factorization: Sparse

Again cyclotomics are a nuisance.

Open Problem ([Roc18, Problem 7])

Suppose $f \in \mathbf{Z}[x]$ *is a* t*-sparse polynomial with at least one sparse factor* $g \in \mathbf{Z}[x]$ *with at most* s *nonzero terms. In polynomial-time in* $t, s, \log H(f)$ *and* $\log \deg f$*, find* **any** s*-sparse factor of* f*.*

Note that "iterate will find **all**", doesn't work, as f/g might be (probably will be) less sparse.

Polynomial Factorization: Sparse, low degree

Here the situation is somewhat better.

Lemma (Gap Lemma [Len99])

"If $f \in F[x]$ can be written as $f = f_0 + x^k f_1$, where the gap $k - \deg f_0$ is large, then every non-cyclotomic low-degree factor of f is a factor of both f_0 and f_1."

This is one of the few "life is better without cyclotomics" genuine theorems

Multivariate Problems

Both [DC10, Roc18] dealt largely with univariates.
A dense polynomial of degree D in n variables has $(D+1)^n$ terms.
While it would be nice to be polynomial in "$t, n, \log D$", I'd settle for "polynomial in t, n, D" (and I expect many system-builders would).
This allows algorithms such as [Zip93] for multivariate g.c.d./factoring.
Though these are still lacking hard analysis, I suspect they are $O(D(\log D)^3)$, and it is probably worth being this detailed

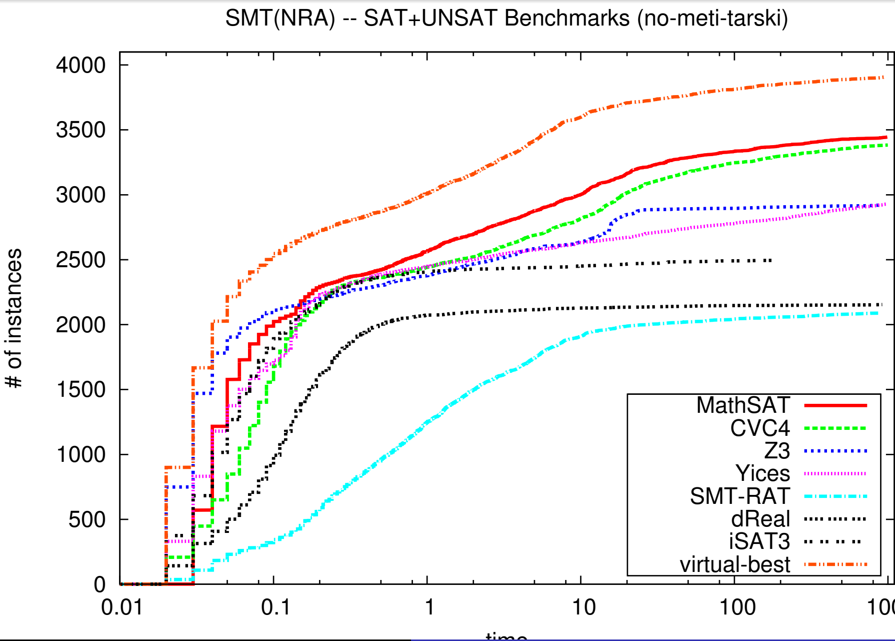
Are there alternatives to complexity theory

The SAT/SMT community could show us some: [Cim18] did.
Major features

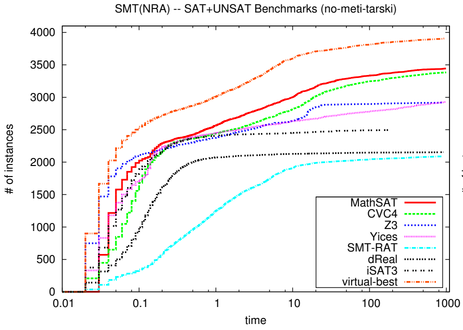
- Lots of examples
- Assume we don't get them all right, and allow a limited time budget
- Include "Virtual Best Solver", i.e. what the per-problem best solver can do

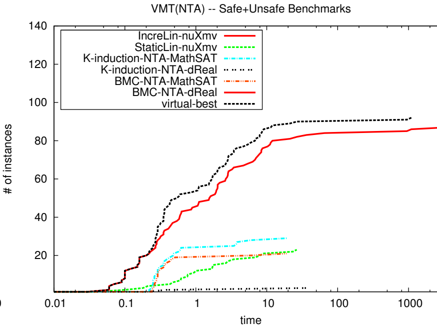
* This is basically "state of the art"

One Graph: [Cim18, Slide 55]



Comparison: [Cim18, Slides 55,73]

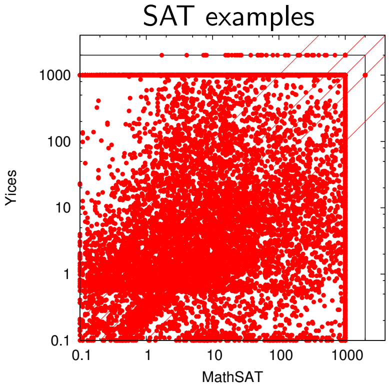


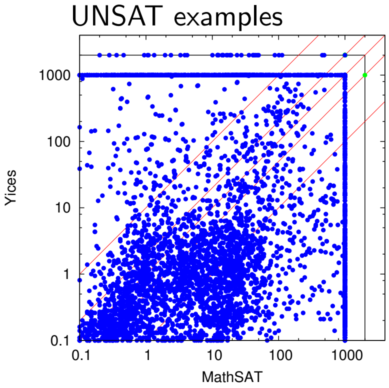


Nearly all solved by VBS	Many unsolved
VBS a lot (?20%) better	VBS scarcely better
Pick up tricks from the others	Look at unsolved problems

Note that 10-100 seconds is fine, which meets the "get a cup of coffee" requirement [BS12]

How to explore: scatter plots [Cim18]





There's a lot of scatter!

On UNSAT, MathSAT is often $\times 10$ faster.

Conclusions

1. Complexity theory *has* served us well
2. But it has its limits, especially in the presence of NP-hardness
3. Other fields (SAT, SMT) handle this differently [BDG17, Cim18]
4. But for this we need large corpora of problems, both “hard” and “typical”
5. And a better (and more honest) benchmarking methodology
6. Contests certainly haven’t hurt SAT
7. But these evolve, unlike “Top 500”, which has hurt HPC [Don16, **?**].

Bibliography I

M.N. Brain, J.H. Davenport, and A. Griggio.
Benchmarking Solvers, SAT-style.
SC^2 2017 Satisfiability Checking and Symbolic Computation CEUR Workshop, 1974(RP3):1–15, 2017.
URL: `http://ceur-ws.org/Vol-1974/RP3.pdf`.

W.S. Brown.
On Euclid's Algorithm and the Computation of Polynomial Greatest Common Divisors.
J. ACM, 18:478–504, 1971.

Martin Brain and Florian Schanda.
A lightweight technique for distributed and incremental verification.
In Rajeev Joshi, Peter Müller, and Andreas Podelski, editors, *Verified Software: Theories, Tools, Experiments*, volume 7152 of *LNCS*, pages 114–129. Springer, January 2012.

Bibliography II

doi:10.1007/978-3-642-27705-4_10.

A. Cimatti.
Incremental Linearization for Satisfiability and Verification Modulo Nonlinear Arithmetic and Transcendental Functions.
Presentation at SYNASC 2018, 2018.

B.G. Claybrook.
Factorization of multivariate polynomials over the integers.
SIGSAM Bulletin, 10:13–13, 1976.

G.E. Collins.
Subresultants and Reduced Polynomial Remainder Sequences.
J. ACM, 14:128–142, 1967.

J.H. Davenport and J. Carette.
The Sparsity Challenges.
In S. Watt *et al.*, editor, *Proceedings SYNASC 2009*, pages 3–7, 2010.

Bibliography III

J.J. Dongarra.
Report on the Sunway TaihuLight System.
http://www.netlib.org/utk/people/JackDongarra/PAPERS/sunway-report-2016.pdf, 2016.

D.E. Knuth.
The Art of Computer Programming, Vol. II, Seminumerical Algorithms.
Addison-Wesley, 1969.

M. Karpinski and I. Shparlinski.
On the Computational Hardness of Testing Square-Freeness of Sparse Polynomials.
In M. Fossorier, H. Imai, S. Lin, and A. Poli, editors, *Proceedings AAECC-13*, pages 492–497, 1999.

Bibliography IV

H.W. Lenstra Jr.
Finding small degree factors of lacunary polynomials.
Number theory in progress, pages 267–276, 1999.

A.K. Lenstra, H.W. Lenstra Jun., and L. Lovász.
Factoring Polynomials with Rational Coefficients.
Math. Ann., 261:515–534, 1982.

D.A. Plaisted.
Sparse Complex Polynomials and Irreducibility.
J. Comp. Syst. Sci., 14:210–221, 1977.

D.A. Plaisted.
Some Polynomial and Integer Divisibility Problems are *NP*-Hard.
SIAM J. Comp., 7:458–464, 1978.

Bibliography V

D.A. Plaisted.
New NP-Hard and NP-Complete Polynomial and Integer Divisibility Problems.
Theor. Comp. Sci., 31:125–138, 1984.

D.S. Roche.
What Can (and Can't) we Do with Sparse Polynomials?
In *Proceedings ISSAC 2018*, pages 25–30, 2018.

A. Schinzel.
On the greatest common divisor of two univariate polynomials, I.
In *A Panorama of number theory or the view from Baker's garden*, pages 337–352. C.U.P., 2003.

Bibliography VI

P.S. Wang.
An Improved Multivariable Polynomial Factorising Algorithm.
Math. Comp., 32:1215–1231, 1978.

R.E. Zippel.
Effective Polynomial Computation.
Kluwer Academic Publishers, 1993.