Digital Collections of Examples in Mathematical Sciences

James Davenport\textsuperscript{1}

University of Bath

9 December 2021

see https://arxiv.org/abs/2107.12908 and [Dav18]

\textsuperscript{1}Partially Supported by EPSRC Grant EP/T015713/1
Plan of Talk

1. Data Citation
2. Important Collections in Pure Mathematics
3. Important Test Suites in SAT/SMT
4. The Lack of Test Suites elsewhere
5. Complexity Theory and its weaknesses
6. Way Forward?
Data Citation

- Is a mess in practice \([\text{vdSNI}^+19]\): only 1.16% of dataset DOIs in Zenodo are cited (and 98.5% of these are self-citation).
- Is poorly harvested: \([\text{vdSNI}^+19, \text{Figure 5}].\)

![Venn diagram showing data sets from Crossref, Europe PMC, and ADS.]

So there are between 4,000–20,000 data sets waiting to be harvested.
Data Citation

- Is a mess in practice [vdSNI+19]: only 1.16% of dataset DOIs in Zenodo are cited (and 98.5% of these are self-citation).
- Is poorly harvested: [vdSNI+19, Figure 5].
- Is still a subject of some uncertainty: [MN12, KS14]
- Changes are still being proposed [DPS+20]
- *de facto* people cite a paper if they can find one.
Important Databases in Pure Mathematics

**OEIS**  Online Encyclopedia of Integer Sequences [Slo03];


But you have to search the website to find it!

+ Large toolset around it.
Group Theory (as an example)

- The Classification of Finite Simple Groups
- The Transitive Groups acting on $n$ points: \[BM83\] ($n \leq 11$); \[Roy87\] ($n = 12$); \[But93\] ($n = 14, 15$); \[Hul96\] ($n = 16$); \[Hul05\] ($17 \leq n \leq 31$); \[CH08\] ($n = 32$).
- These are in GAP (and MAGMA), except that $n = 32$ isn’t in the default build.
  - These are a really great resource (if that’s what you want)
  - How do you cite them? “[The21, GAP transgrp library]”?
  
Also

- Other libraries such as primitive groups

Group Theory is “easy”: for a given $n$ there are a finite number and we “just” have to list them.
SAT Solving

SAT solving, normally seen as solving a Boolean expression written in CNF. Given a 3-literals/clause CNF satisfiability problem,

\[(l_{1,1} \lor l_{1,2} \lor l_{1,3}) \land (l_{2,1} \lor l_{2,2} \lor l_{2,3}) \land \cdots \land (l_{N,1} \lor l_{N,2} \lor l_{N,3}),\]

where \(l_{i,j} \in \{x_1, \overline{x_1}, x_2, \overline{x_2}, \ldots\}\), is it satisfiable? In other words, is there an assignment of \(\{T, F\}\) to the \(x_i\) such that all the clauses are simultaneously true.

3-SAT: the quintessential NP-complete problem [Coo66]. 2-SAT is polynomial, and \(k\)-SAT for \(k > 3\) is polynomial-transformable into 3-SAT. In practice we deal with SAT — i.e. no limitations on the length of the clauses.

Let \(n\) be the number of \(i\) such that \(x_i\) (and/or \(\overline{x_i}\)) actually occur. Typically \(n\) is of a similar size to \(N\).
Despite being NP-complete, nearly all examples are easy (e.g. [KS00] for the automotive industry), either easily solved (SAT) or easily proved insoluble (UNSAT) and for random problems there seems to be a distinct phase transition between the two: [GW94, AP04, AP06].
This means that constructing difficult examples is itself difficult, and a research area in itself: [Spe15, BC18].
SAT solving has many applications, e.g. new ways of multiplying matrices [HKS21], so we want effective solvers for “real” problems, not just “random” ones.

Fundamental question: what does this mean?
SAT Contests: http://www.satcompetition.org

Been run since 2002. In the early years, distinct tracks for Industrial/Handmade/Random problems: this has been abandoned. The methodology is that the organisers accept submissions (from contestants and others), then produce a list of problems (in a standard format) and set a time (and memory) limit, and see how many of the problems the submitted systems can solve on the contest hardware.

SAT is easy to certify (just produce a list of values), UNSAT is much harder, but since 2013 the contest has required proofs of UNSAT for the UNSAT track, and since 2020 in all tracks, in DRAT: a specified format (some have been > 100GB).

The general feeling is that these contests have really pushed the development of SAT solvers, roughly speaking $\times 2$/year. For comparison, Linear Programming has done $\times 1.8$ over a greater timeline [Bix15], chips $\times 1.41$ [Moo75].
SAT in practice

Although SAT, and \( k \)-SAT, are reducible to 3-SAT, we shouldn’t do this in practice. Indeed, some solvers look for this reduction and undo it. A significant practical development was “Two Watched Literals” [MMZ\(^+\)01], which means that, most of the time, we don’t look at a whole clause.

*In the worst case it is no more efficient than the algorithm it replaced, in practice it is hundreds of times more efficient.*

The “not moving back” feels like it should save at most half the time, it in fact speeds things up by often 10 times. [https://news.ycombinator.com/item?id=18236555](https://news.ycombinator.com/item?id=18236555)

In particular it is much more cache-efficient on modern processors. This is essentially orthogonal to complexity theory as we know it.
Consider a theory $T$, with variables $y_j$, and various Boolean-valued statements in $T$ of the form $F_i(y_1, \ldots, y_n)$, and a CNF with $F_i(y_1, \ldots, y_n)$ rather than just $x_i$.

Then the SAT/UNSAT question is similar ($\exists$ values of $y_i$ ...), and the community runs SMT Competitions (https://smt-comp.github.io/2020/), but a separate track for each theory, as the problems will be different.

The SMTLIB format [BFT17] provides a standard input format. UNSAT is in general unsolved (but see [KAED21] for one example). There is substantial progress in SMT-solving over the years, possibly similar to SAT.
Computer Algebra: where are we

Obviously, Group Theory (etc.) are part of computer algebra: what about the rest?

Much traditional complexity theory is for dense polynomials/matrices/. . . , but real life is often about sparse ones.

*In general* the problems have a bad worst-case complexity, and we want effective solvers for “real” problems, not just “random” ones.

The question is “what does this mean?”.

**Format**

No common standard. We do have OpenMath [BCC+17], but it’s not as widely supported as we would like.

**Contests**

None. Could SIGSAM organise them?

**Problem Sets**

No independent ones. Each author chooses his own.

**Archive**

Not really.
Polynomial g.c.d.

- NP-hard (for sparse polynomials, even univariate) [Pla84].
- Can be challenging for multivariates
  - No standard database: trawl previous papers (and often need to ask the authors)
- Verification is a challenge: one can check that the result is a common divisor, but verifying greatest is still NP-hard [Pla84].
The NP-hardness results of [Pla84] rely on encoding a SAT-formula $W$ in $x_1, \ldots, x_n$ as $P_M(W)$, which vanishes at the $M = \prod_{i=1}^{n} p_i$-th roots of unity corresponding to satisfying assignments for $W$. There are also blow-up results [Sch03]

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

[DC10] asks whether these problems are limited to cyclotomics, and [CD10] asks about explicitly encoding cyclotomics, so this would be $\gcd(C_{pq}, C_p C_q) = \frac{C_p C_q}{C_1}$. Both are open problems.
Polynomial Factorisation

1. Standard algorithm [Zas69] has exponential worst-case behaviour [SD69].
2. [Col79] conjectured polynomial average*-time.
3. Polynomial-time for dense encodings [LLL82], but a very bad exponent.
4. Exponentially larger output for sparse encodings:
   \[ x^p - 1 = (x - 1)(x^{p-1} + \cdots + 1) \]
5. Presumably NP-hard (even in terms of output size) for sparse.

Verification is a challenge: one can check that the result is a factorisation, but checking completeness (i.e. that these factors are irreducible) seems to be as hard as the original problem (in the worst case).
With probability 1, a random polynomial is irreducible, so what are the *interesting* problems?

* Therefore [Col79] conjecture needed a subtle definition of “average” time

* [ABD88] points out that “difficult” examples arise from factoring over algebraic number fields, which happen in “real life”.

Hence An “engineering” task of switching between “exponential but generally quick” and “slow but guaranteed polynomial”.

- No standard database: trawl previous papers (and often need to ask the authors)

So what constitutes a good benchmark for such tasks?
Gröbner Bases

- Doubly exponential (w.r.t. $n$) worst-case complexity [MR13], even if a prime ideal [Chi09].
  - There is a collection [BM96]
    - Very old (1996) and completely static.
    - Some examples only in PDF.

Again  The generic case is not interesting: Shape Lemma [BMMT94].

?  No concept of UNSAT, but it’s not clear what a certificate might mean.
Real Algebraic Geometry (CAD)

- Doubly exponential (w.r.t. $n$) worst-case complexity [BD07]

+ There is a collection [Wil14]
- Somewhat old (2014) and completely static.

✓ The DEWCAD project [BDE+21] might update this, but still issues of long-term conservation.

❓ Format: text, Maple and QEPCAD

❓ No concept of UNSAT (but see [KAED21]), and it’s not clear what a “certificate” might mean.
Let $a$ be the number of alternations of quantifiers.

[BD07] has $a = n - 1$.

There are (unimplemented) algorithms for quantifier elimination which are $d^{\text{poly}(n)2^a}$, and implemented algorithms which might have similar complexity.

The world of software, or cyber-physical systems, tends to produce examples with $a = 0$, i.e. $\exists \text{geometry} : \text{bad(geometry)}$.

Are there natural examples for quantifier elimination with a non-trivial?
Integration

- Complexity is essentially unknown (but certainly involves g.c.d., factorisation etc.)
- A new question here is the “niceness” of the output.
- “Paper” mathematics produced large databases, e.g. [GR07].
  - PDF, and the devil to scan.
- Current best database is described in [JR10].
- Algorithm-based software (e.g. [Dav81]) has an internal proof of UNSAT, but I know of no software that can exhibit it.
“Average-case complexity without the black swans”: i.e. without an exponentially-rare family that is worse than exponentially bad.

**Definition**

For $k \in \mathbb{N}$ let $(M_k, \mu_k)$ be a probability space and let $T_k : M_k \rightarrow \mathbb{R}$ be a $\mu_k$-measurable function. We say that the family $\{T_k\}$ has a weak expectation of $O(f(k))$ if there exists a family of sets of exceptional inputs, $E_k \subseteq M_k$, such that $\mu_k(E_k) = e^{-\Omega(k)}$ and the conditional expectation $E[T_k(x) | x \notin E_k]$ is bounded by $O(f(k))$.

- Condition numbers inversely proportional to a distance to a homogeneous algebraic set of ill-posed inputs;
- Renegar’s condition number for conic optimization;
- The running time of power iteration for computing a leading eigenvector of a Hermitian matrix.
Guesses

**Guess**

Gröbner basis computation is weakly singly exponential in $n$.

Needs a measure on sets of polynomial inputs, or possibly a measure on ideals. Might need some further restrictions e.g. to radical ideals (but see [Chi09])

**Guess**

Real Quantifier Elimination is weakly singly exponential in $n$ for any $a/n$ ratio.

Needs a measure on inputs, I think.
Conclusions

1. We need to be more inventive around complexity theory.

2. The field of computer algebra really ought to invest in the sort of contests that have stimulated the SAT and SMT worlds.

3. This requires much larger databases of “relevant” problems than we currently have, and they need to be properly curated.

4. Technology, e.g. wikis, or GitHub, has greatly advanced since [BM96].

5. These collections would allow much better benchmarking technology [BDG17].


D. Achlioptas and Y. Peres.
Random $k$-SAT: Two Moments Suffice to Cross a Sharp Threshold.

T. Balyo and L. Chrpa.
Using algorithm configuration tools to generate hard SAT benchmarks.
URL: https://algo2.iti.kit.edu/balyo/papers/socs18.pdf.

The DEWCAD Project: Pushing Back the Doubly Exponential Wall of Cylindrical Algebraic Decomposition.

M.N. Brain, J.H. Davenport, and A. Griggio.
Benchmarking Solvers, SAT-style.
E.R. Berlekamp.
Factoring Polynomials over Large Finite Fields.

C. Barrett, P. Fontaine, and C. Tinelli.
http://smtlib.cs.uiowa.edu/papers/

R.E. Bixby.
Computational Progress in Linear and Mixed Integer Programming.
*Presentation at ICIAM 2015*, 2015.
URL: http://staff.bath.ac.uk/masjhd/Others/Bixby2015a.pdf.
G. Butler and J. McKay.
The transitive groups of degree up to 11.

D. Bini and B. Mourrain.
Polynomial test suite.

E. Becker, M.G. Marinari, T. Mora, and C. Traverso.
The shape of the shape lemma.

G. Butler.
The transitive groups of degree fourteen and fifteen.
Bibliography VII

J. Carette and J.H. Davenport.
The Power of Vocabulary: The Case of Cyclotomic Polynomials.

J.J. Cannon and D.F. Holt.
The transitive permutation groups of degree 32.

A.L. Chistov.
Double-exponential lower bound for the degree of any system of generators of a polynomial prime ideal.
Bibliography VIII

G.E. Collins.
Factoring univariate integral polynomials in polynomial average time.

S.A. Cook.
*On the minimum computation time of functions.*

J.H. Davenport.
J.H. Davenport.
The Role of Benchmarking in Symbolic Computation.
URL:
https://researchportal.bath.ac.uk/en/publications/the-role-of-benchmarking-in-symbolic-computation-position-paper,
doi:10.1109/SYNASC.2018.00050.

J.H. Davenport and J. Carette.
The Sparsity Challenges.


M.J. Heule, M. Kauers, and M. Seidl.

New ways to multiply $3 \times 3$-matrices.


A. Hulpke.

*Konstruktion transitiver Permutationsgruppen*.


A. Hulpke.

Constructing transitive permutation groups.


D.J. Jeffrey and A.D. Rich.

Reducing Expression Size Using Rule-Based Integration.

On the Implementation of Cylindrical Algebraic Coverings for Satisfiability Modulo Theories Solving.

W. Küchlin and C. Sinz.
Proving Consistency Assertions for Automotive Product Data Management.

J. Kratz and C. Strasser.
Data publication consensus and controversies (version 3).
Bibliography XIII


Bibliography XIV

G.E. Moore.
Progress in Digital Integrated Electronics.

E.W. Mayr and S. Ritscher.
Dimension-dependent bounds for Gröbner bases of polynomial ideals.

D.A. Plaisted.
New NP-Hard and NP-Complete Polynomial and Integer Divisibility Problems.
G.F. Royle.
The Transitive Groups of Degree Twelve.

A. Schinzel.
On the greatest common divisor of two univariate polynomials, I.

H.P.F. Swinnerton-Dyer.
Letter to E.R. Berlekamp.
*Mentioned in [Ber70], 1969.*
Bibliography XVI

N.J.A. Sloane.
The Online Encyclopedia of Integer Sequences.

I. Spence.
Weakening Cardinality Constraints Creates Harder Satisfiability Benchmarks.

The GAP Group.
Practice meets Principle: Tracking Software and Data Citations to Zenodo DOIs.

D.J. Wilson.
Advances in Cylindrical Algebraic Decomposition.

H. Zassenhaus.
On Hensel Factorization I.