ISSAC/PASCO 2015

J.H. Davenport — J.H.Davenport@bath.ac.uk

July 2015

Contents

Chapter 1

6 July 2015

[AR14] [Arn03] [BDEW13] [Bro71] [Dav85] [ESY06] [Fau99] [FM13] [FSEDS10] [GR11] [JdM12] [Len99a] [Len99

1.1 An Introduction to Finite Element methods: Veronika Pillwein

A numerical method for finding approximate solutions to PDEs on non-trivial domains. Divide the domain into geometrically-simple subdomains, and the solutions are represented by locally-supported polynomials. objects. Uses NetGen 4.5 for the mesh generation.

1.1.1 1D example

The bound is quite short and nice, but the function being bounded is several screens. This is partly because we are, dealing, separately, with two pre-smoothing and two post-smoothing steps. See `http://www.risc.jku.at/people/vpillwei/sFLA` — nice conjecture, but no proof.

Q Which CAD?

A Mathematica. Four variables, of degree ≤ 10.

Q The best bound?

A This is an upper bound because of the separation of the two steps. But I do know that it is achieved.

Chapter 2

7 July 2015

2.1 Ábráham

2.1.1 SAT problems

Just Booleans with no underlying theory.

Resolution: what JHD would think of as elimination. But this would have combinatorial blow-up.

Enumeration: try a true, then $\neg a$ true, within each, try $b, \ldots$. Again this has exponential complexity.

SAT Solving the DPLL algorithm.Decision, see if variables occor only positively or only negatively. If both, decide which (essentially enumeration). Then Propagate the consequences of this decisions, look for a conflict, then backtrack.

Next breakthrough is conflict-directed clause learning (CDCL). Roughly, use resolution to analyse a conflict. Discover that $(\neg a \vee \neg d \vee \neg e)$ and $(\neg a \vee \neg d \vee \neg e)$ combine and give a new clause $\neg a \vee \neg d$.

2.1.2 Satisfiability Modulo Theories

Propositional logic is sometimes too weak. SMT-LIB has been a standard input language since 20904. Competitions since 2005. SMP-COMP 2014 had 32 logic categories and 20 solvers. Linear R had 6 solvers, non-linear R had 4. 67426 benchmark instances. Examples (in order of increaing expressivity) Array logic, quanti

er-free integer/rational difference logic, quantiffer-free linear arithmetic; qf nonlinear arithmetic. Quantifier-free FO formula goes via Boolean abstraction and Tseitin's transformation to a propositional CNF. This goes into a SAT solver, which interacts worth the theory solver. The theory solver feeds back SAT/UNSAT and lemmas.

Example 1 $((x < 0) \vee (x > 2)) \wedge (x^2 = 1 \vee x^2 < 0)$. *This then becomes* $(a \vee b) \wedge (c \vee d)$. *The underlying theory solver might feed back* $\neg(x^2 < 0)$, *which is treated as a new clause (JHD: rather like CDCL).*

Gröbner bases, CAD, Virtual Substitution are already implemented in CAS. But these are not SMT-compliant, i.e.

- work incrementally
- generate lemmas explaining inconsistencies
- be able to backtrack

Current implementations are not that, probably not available as libraries, and generally not thread-safe. Usually, SMT-adoption is not trivial.

Example 2 *SMT-RAT library of theory modules.* *`https://github.com/smtrat/smtrat/wiki`* *Has interval constraints, Simplex algorithm, Virtual substitution [FCT11, PhD Corzilius] CAD [CADE24, PhD Loup, . . .]. Gröbner bases, but only for simplification, not for true solving.*

2.1.3 Virtual Substitution

The key idea is to replace an existential quantifier in favour of a finite disjunction over parametric set points. $\exists x, y : (y = 0 \vee y^2 + 1 < 0) \wedge x - 3 \leq 0 \wedge xy + 1 < 0$ Eliminate x gives tests $x = -\infty$, $x = 3$, $x = -\frac{1}{y} + \epsilon$. Then we have $\exists y$. Fail by failing $y^2 + 1 < 0$, and this is independent of x, so the SAT-solver learns this new clause.

Claims that this is not dissimilar from CAD, where we have projection/construction as structurally similar. Various benchmark problems, often showing her sytem best, but also z3 does best on the Metitarski test set.

Notes that the two communities are very disjoint: different journals, conferences, tools and goals.

2.2 Software Snippets

2.2.1 Cloudmath

Idea is to use an interface with HTML5/GSS/javascript, whhc talks to the CloudMathEngine and Plot server, and this talks as web services to Python, SAGE, Octave, R etc.

2.2.2 Lacunary

The best factorisation (polynomial) algorithms we know are polynomial in $\deg(f)$. Want objects in $\log \deg f$ [Cuckeretal1998] [Len99a, Len99b] [KK05] and more recent, for sparse polynomials. Note the split cyclotomic/noncyclotomic factors. [Len99a, Len99b] has gap theorems, which says that factors of degree

$\leq d$ must divide each part (split by gaps $> \gamma(f, d)$). Have various improvements on this. Can handle polynomials of degree $\geq$ 1M in under three minutes.

2.2.3 Abbott

CoCoA founded by Robbiano in 1987, an=imed as a mathematician-friendly tool. Gröbner bases, Hilbert series, ideals of points, more recently approximate points, border bases etc. Three interfaces: CoCoaLib as a C++ software library with GMP-3 licence, the interpreter CoCoA-5, and an OpenMat-based prototype server. Based on GMP.

CoCoA-5 Natural dynamically-typed syntax. Allows new types of rings (algegraic extensions) and speaks about ring homomorhisms.

2.3 Mora

Claims that you can build a Buchberger theory for effective associate rings. [**?**].

Theorem 1 *For an (associate but not necessarily commutative) ring with identity A, there is ....*

Se let A be a left module over R, and it is effetcive if we are given sets $\overline{v} = \{x_1, \ldots, \ldots\}$ and $\overline{V} = \{X_1, \ldots, X_m, \ldots\}$ such that ....

Then $\Pi : \mathbf{Z}\langle < x_i, X_j\rangle \to A$ and $\pi : R := \mathbf{Z}\langle x_i\rangle \to R$. Termination requires an additional condition.

$$X_i x_j = \sum_{l=1}^{i} \pi(a_{lij} \ldots$$

and therefore $X_i x_j$ can't appear as a head term, as it's reducible.

Theorem 2 (Zacharias's Theorem) *[BScTheis MIT1978] Canconical forrms A_c $\mathbf{Z}(\overline{Z})^m/I \equiv Zach(\mathbf{Z}(Z))\ldots$.*

Theorem 3 (Spear's Theorem) *[Macsyma1977]. Tells us that a Buchberger theory defined in a ting can be exorted to its quotients which allows us to impose on A the "naural" Γ-valuation/filtrattion*

$$T(\cdot) : A^m \mapsto B^{(m)} : f \to T(f).$$

The Γ-graded ring $G = G(A)$ coincides as a set with A, which is sufficient to export Buchberger test/completion, but they don't coincide as rings, as the multiplications don't coincide. However, an old slogan says that inorder to proide a Buchberger Algorithm on A, one just takes the one for G and modifies the multiplication method.

Theorem 4 (Möller's Theorem) *[JSC1988] This says the same things (about lifting) in a diferent way. A basis F is Gröbner iff each element u ina a minimla absis of the mpdule $\ker(s)$ of the syzygies among the leading monomials $M(g_i)$ lifts, via Buchberger reduction on $G(u)$ to a syzygy $U \in \ker(G)$ among the g_i. A corollary of this is Janett–Scheier theorem that the lifted elements form a Gröbner basis of $\ker(G)$.*

2.4 Sturm

Hpof bifurcations in system biology and chemistry. Example of a gene regulated bya protein. A quisi-steady state approximantion gives a reduced system. Boulier et al. examine the existence. Maple implementatoin by El Kahoiui/Weber, ϕ_n holds for some choice of parameters iff a Hopf bifurcation exists. First consider $\exists\phi_n$. Note that all variables are strictly positive (positive QE). [CASC2013] for the Methylene Blue Oscillator: one large eequation, side small-equations, and positivity condition. Decompose the large problem into 500 sub-problems. Does any one of these have a positive solution?

A still larger probem is MAPK [J. Comp. Phys. 2015] MBO is 7 variables, degrees 4–9, 6000 monomials. This is 10/5–12,863000 monomials (30MB). This takes 10s for finding positive/negative point, and 5s for exact solving. Reduce takes 25s for reading/parsing. 2s to find "degree". Size limit of 32MB in Maple (used in differential lagebra part of chain) polynomials becomes a constraint. Let $\mathrm{supp}^+(f)$ be the support with positive coefficients, and supp^- similarly. Once we have a positive point and a negative one, can join witha line and use intermediate value theorem.

2.5 Real root finding . . . : Naldi

A linear ,matrix is a polynomial matrix of degree 1. $A(x) = A_0 + x_1A_1 + \cdots + x_nA_n$. Suppose the $A_i \in |q^{m\times m}$ and are Hankel. Lyaponov stability if a Limear Matrix Inequality (LMI). Hence a spectrahedron. ≥ 0 means "positive semi-definite".

The Hankel spectrahedra are integresting objects, include the Carathéodory orbitope. Consider the loimear actio of $SO_2(\mathbf{R})$ on the space of homogeneous quartics, then the convex hull of the orbit of $F(X,Y) = X^4$ is

$$\begin{pmatrix} x_0 & x_1 & x_2 \\ x_1 & x_2 & x_3 \\ x_2 & x_3 & x_4 \end{pmatrix} \geq 0; \qquad x_0 + 2x_2 + x_4 = 0$$

Let $D_r = \{x \in \mathbf{C}^n | \operatorname{rank}(A(X) \leq r\}$. Problem: compute one point in each connected computent of $D_r \cap \mathbf{R}^n$.

We use FGb to compute the Gröbner bases, Comparsion's with Safey El Din's RAGLIB,

2.6 Near Optimal Subdivision Algorithms for Real Root Isolation: Sharma

Subdivision methods: two predicates $C_0(I)$ says "there is no root", $C_1(I)$ says "exactly one root". These have a variety of instantiations. Note that these are partial: might say "no" even though the condition is true. Consider $\#T$, the size of the tree, and the worst case arithmetic complexity at a node, which is normally $\widetilde{O}(n)$. Typical bound on $\#T$ is $O(\log 1/\sigma)$ where σ is the root separation bound. See [Dav85], [ESY06, SY12].

Subdivision plus Newton iteration is our remedy for clustering. There's an annulus around the cluster where this converges quadratically to the median of the cluster.. See [Pan00] for predicates based on distance ot nearest root. citeSagraloff2012a which has quadratic refinement..

Our result works for any instantiations. $O(n \log n)$ for Descartes C_0 or Sturm's C_1. Analysis is independent of root bounds: uses geometry of clusters. If $C \cap C' \neq \emptyset$, then either $C \subset C'$ or *vice versa*. Ordinary clusers are found by bisection, strongly separated clusters by Newton.

Use continuous amortization, Stopping function: given J, if $\exists x \in J$ such that $w(J)G(x) \leq 1$ then either $C_o(J)$ or $C_1(J)$ holds.

2.7 Open Non-Uniform Cylindrical Algebraic Decomposition: Brown

First exposure of this new idea, even though CAD is old. Tarski formulae provide *implcit* representations of semi-algebraic sets: not very helpful in general. Hard to do `isempty`, for example. "Open cell" generalised the concept of a bax aligned with the axes, x is still given as $l < x < r$, but y is between non-intersecting algebaric functoins of x, x is terms of x, y etc. So an Open CAD is a weak decomposition into these cells, which is cylindrical in the sense that

$$\pi_k(C_i), \pi_k(C_j) \text{ are equal or disjoint.} \tag{2.1}$$

Down in $\mathbf{R}^1$, pick a sample point in each open interval. Then lift and so on. We do a lot of work before we ever reach a sample point in $\mathbf{R}^n$. [JdM12] given a point α and a formala F constructs a cell containing α in which F has constant truth value. See [**?**]. This introduced `merge`: given a cell $c \ni \alpha$, and polynomiaal p, construct $c' \subseteq c$ containing α such that p has constant sign on c'.

Aim is to discard (2.1). Start with the empty set of constraints, i.e. $\mathbf{R}^n$, and a random sample point. Keep `merge`, or move to an excluded cell if p has the wrong sign. Far fewer cells and far fewer decision points in lower dimensions. This is in some sense an adaptive algorithm. Has a good parallelism model as well.

1. Q a queue containg $\mathbf{R}^n, 0$
2. while Q nonempty
   (a) Take a cell off the queue
   (b) Add a constraint ...

Example with linear polynomials. 5 in 4 variables has 233 cells, versus 50K with QEPCAD. Nonlinear (slightly) 4 polynomials/4 variables has 4147 cells rather than 447K,

Q Does this only handle conjunctions.

A Yes, for reasons of implementation.

Q Reuse?

A Yes, one has to check this. But the number computed (if you do this) is far fewer than with straight CAD.

Q Complexity?

A No idea yet

Q For QE/CAD you need cylindricity?

A Yes, but these cells are relatively easy to project, and we need to resolve overlaps.

Q (JHD) this is block-cylindrical.

A Yes.

2.8 Improving the Use of Equational Constraints: England

Note Mccallum 1998/1999. 1998 is order in, order out, but 1999 $P_F(B)$ is order in/sign out. Hnece 1999 doesn't compose, and only supports one equational constraint. McCallum 2001 does better. Note all these improve projection.

Lifting is nevertheless the most expensive stage. Hence we can lift only with respect toequationaal contrraints (which involves discarding the idea that all polynomials are equal).

The second idea is that our decomposition in a loer varable equationally constrained consists of sectors (hence the EC can't be true) and sections. The secotors can just be lifted over trivially. Note that the resulting CAD will be truth-invariant for the formula, but it may not be sign-invariant for any one polynomial.

Example 3 *Five variable,s but four equational contraints. We have 60 different routes for designation, but only 113, 103 or 93 cells (three real possibilities), but a sign-invarant CAD is* $> 1M$ *cells, and one order-invariant is 150K or 11K. Propragating all ECs gives 21K by default, and the correct designation can make 5633. Reular Chains does produce 137 cells.*

We analyse #cells. m polynomails of degree d in n variables. Assume t equations. The degree-dependence doesn't change. We replace m^{2^n} by $m^{2^{n-t}}$, roughly speaking.

Note that this only works with primitive equational constraints. The problem with contents is that they are a disjunction, and it's no longer true that constrints are nonzero on sectors.

Q–TS Note that VTS makes a non-degeneracy assumption, which essentially ignores equational constraints. I also feel that QE via CAD suffers from the problem that it answers all questions.

A Indeed. [BDEW13] is going some way to solve this.

2.9 Real Quantifier Elimination by Computation of Comprehensive Gröbner Basis: Sato

Motivation is the Todai robot project. In GCS-QE we use the Real Root Counting Theorem (Pedersen). $\sigma(M_p^I) = \#(\{\bar{c} \in V_R(I)|p(\bar{c}) > 0\}) - \#(\{\bar{c} \in V_R(I)|p(\bar{c}) < 0\})$.

Add polynomials $z_i^2 p_i - 1$, where z_i is a new variable. Then we can count solutions with $p_i > 0$, but each z_i gives a factor of 2 we need to remove (since $\pm z_i$).

In an example, the original GCS-QE would have polynomials of degree 24, whereas we have degree 8.

We compute the saturation ideal I' of U with respect tothe p_i, and this reduces the dimension. We can also use a primary decomosition of I to remove unecessary portions from I.

Have a concept of algebraic partitions if

1. union in S
2. $S_i \cap S_j = \emptyset$ if $i \neq j$.
3. $S_i = V_c \ldots$

We allows equations f_i and inequalities p_i and inequations q_i. Let M be a real symmetric matrix and $\chi(X)$ be its characteristic polynomial, with coefficients a_i. Let b_i be the copefficients of $\chi(-X)$, so $b_i = \pm a_i$.. As in RRC Theorem, $\#(V_{\mathbf{R}}(I)) = \sigma(M_I^I)$.

1. Introduce new variables $|_i, W_j$.
2. Term order with these $> X - k$.
3. Consider $f_i, Z_j^2 p_j - 1, W_k q_k - 1$ as the new polynomials
4. $\ldots$

Future work:

Q-ME This is in SYNRAC: is it available?

A Apparently yes

Q-TS You are assuming existential quantifiers and DNF. In general this is not primitive recursive.

A For Todai examples this seems to work.

2.10 Business Meeting: MK chaired

2.10.1 New members to replace MK

- Dan Roche. I would like to consider what (small) changes we could make to ISSAC.
- Frédéric Chyzak. Been in the community since 1990s.

2.10.2 Bids

Kaiserlslautern Decker *et al.* In the middle of Pfälzer Wald. 90 minutes from Frankfurt airport, or 150 from Paris. Smalluniversity with strng mathematics. Home of SINGULAR, group of about 40. Looking at a registration of 250euros plus a banquet of 50. B&B rooms start at 54euros, and there's an even cheaper hostel. The mensa on campus is very efficient.

Q-JHD Inthe past people have said banquets should be inclusive.

A Noted

Q Dates?

A July sometime.

Jilin University, Changchun = "Long Spring" Prof Na LEI, Shugong ZHANG. Have hosted three previous conferences. CM2013, NCA2015 (500 participants). Suggest July 24–27 including tutorials. Was the capital of Manchkou 1932–45, also PuYi's last residence. Averages 23C, but rainfall is heaviny concentrated in June–August. Jilin has 8 campus and is rated Number 8 in all Chinese universities. Many flights from China, also Seoul, Tokyo, OSaka. Suggest using alarge local hotel with excellent conference facilities. Suggest $300 regular, $150 student. Senior researcher accommodation $100–120/night, students $80.

2.10.3 ISSAC 2015

74 submissions, 43 acceptances. At the low end of recent years. JSC special issue: call probably in November.

16 people pre-registers: 30 SIGSAM, 31 Students, 49 non-SIGAM, 6 invited. Now looking at 128. Only 19 people have ordered the paper proceedings, so we have a few spare. Note that ACM takes 16%, and require a 7.5% contingency (15% is the norm, so thanks for SIGSAM). Thanks also to Maplesoft and the London Mathematical Society. Expect to make a small profit.

2.10.4 ISSAC 2016: Ilias

ISSAC 2016 at WLU in Waterloo. July 20–22. Tutorials before. Chairs Abramow and Zima. PC Chair Xiao-Shan Gao. Treasurer Jacques Carette. Proceeedings Markus Rosenkranz. We will be in the new Global Innovation

Exchange building at WLU. Due to be finished in September. Fields Institute being approached for sponsorship, also Perimeter Institute.

2.10.5 Awards

It has been suggested that in future the Fachgruppe sponsors the Best Poster/Software awards when in Europe, and SIGSAM when not in Europe. JAA also noted that the PC had problems with the best student award, since "senior authors" wrote unhelpful comments.

2.10.6 Results

Roche beats Chyzak 39–36. Kaiserslautern beats China 59–20.

2.11 SIGSAM Business—IK

Thanks to Matthew England for the new SIGSAM website.

Notes how important it is that CCA is referenced in Scopus etc.

Treasurer's report. Note that the \$10K SIGSAM pays ACM is (slightly) less than the SIGSAM-related (ISSAC+CCA) revenue. Therefore the conference charge etc. stays within SIGSAM.

Q Why not give some money back to ISSAC?

A A few years ago (2008) we were below ACM's rules for minimal balances, so this wasn't feasible.

JHD A small support fund would have been very valuable, say \$3000 (I do mean \$!). **JHD to write proposal**. General Approval.

Chapter 3

8 July 2015

3.1

Consider a basic semi-algebaric function, and optimising a linear function over it. Polyhedron is linear programming. Consider the semi-definite representation of the closure of the convex hull of S.

Also interestied in optimising a parametric linear function over a real alebaric variety. $c_0^* = \sum_{xinS} c^T x$ for unspecified

3.1.1 Semidefinite representations

Not ethat

$$\sup_{x \in S} c^T x = \sup_{x \in cl(co(S))} c^t x$$

i.e. we are only interested in teh convex part (because linear!).

$S \subset \mathbf{R}^n$ is a *spectrahedron* if it is

$$S = \{(x_1, \ldots, x_n) \in \mathbf{R}^n | A_0 + \sum A_i x_i \succeq 0\}$$

where the A_i are given symmetric matrices.

Note that the TV set is not a spectrahedron, but is a projected spectrahedron.

Conjecture 1 (HelotonNie) *Every convex semialgebaric set has a semidefinite representation.* Proved Schneiderer for $n = 2$

Can we characterise $p|_S \geq 0$ by sums of squares of polynomials. Let Σ^2 be the set of SOS polynomials.

Definition 1 *The quadratic mudole generated by* $G = \{g_i\}$ *is*

$$Q(G) := \{\sigma_0 + \sigma_1 g_1 + \cdots + \sigma_m g_m\}$$

where $\sigma_i \in \Sigma^2$. $Q_k(G)$ *when* $\deg(\sigma_j g_j) \leq 2k \ \forall j$.

So $p \in Q(G) \Rightarrow p_S \geq 0$: when is the converse true? Suppose $Q(G)$ satisfies the *Archimedean condition.*

Definition 2 *The kth theta body of G is*

$$TH_k(G) := \{x \in \mathbf{R}^n | p(x) \geq 0 : \forall p \in Q_k(G) \cap \mathbf{R}[X]_1\}.$$

$TH_1(G) \supseteq TH_2(G) \supseteq \cdots$. Putinir's Positivstellensatz implies ....

Lasserre's Semidefinitine Relaxations of $cl(co(S))$. Given $y = \{y_\alpha\}$, let $L_y : \mathbf{R}[X] \to \mathbf{R}$ bethe linear functions

$$L_y(\sum_\alpha q_\alpha X^\alpha) \mapsto \sum_\alpha q_\alpha y_\alpha.$$

Moment marix

$$M_k(y)(\alpha, \beta) := L_y(X^\alpha X^\beta) = y_{\alpha+\beta}.$$

Then definte kth Lasserre relaxation, and when $Q(G)$ is Archimedean, the the dual side of Putain implites

$$cl(co(S)) = \bigcap_{k=1}^{\infty} cl(\Omega_k(G)).$$

3.1.2 S not compact?

Shows an example where Ω_k is very different (too large) and doesn't help.

- Homogenisation

$$\tilde{f}(\widetilde{X}) = X_0^{\deg(f)} f(X/X_0) \in \mathbf{R}[\widetilde{X}]$$

- Lift S to a cone $\widetilde{S}_i$ Then $f(x) \geq 0$ in $S \Leftrightarrow \tilde{f}(\tilde{x}) \geq 0$ on $cl(\widetilde{S}_1)$.

We say S *is closed at* ∞ of $cl(\widetilde{S}_1) = \widetilde{S}_1 = \ldots$. This gives us a modified Lasserree hierarchy $\widetilde{\Omega}_k(\widetilde{G})$ and modified Theta body.

3.1.3 Pointed Convex Cone

Closed and contains no lines. So assume S is closde at infinity and the convex cone is pointed. Then [GuoWangZhi]

$$cl(co(S)) \subseteq cl(\tilde{\Omega}_k(widetildeG)) \subseteq \widetilde{TH} \cdots$$

For pointed but not clsoed at infinity, then we get an example with $\widetilde{TH}_k = \mathbf{R}^2 \neq$ what it should be.

But "closed at infinity" depends on the actual generators of S, so is not intrinsic.

Also shows that lack of pointedness might lead to non-convergence.

3.1.4 Parametric linear functions

Tarski–Seidenberg implies that the optimalvalue function is semi-algebaric. But how do we compute $\Phi \in \mathbf{R}[c_0, \mathbf{c}]$. CAD, but limited to small n. KKT equations for $\mathcal{V}$ being irreducible, smooth and compact in $\mathbf{R}^n$ [Rostalski,Sturmfels].

The dual Variety V^* is the Zariski closure of the set

$$\{u \in \mathbf{P}^n | u \text{ lies in the row space of} Jac(V) \text{ at } x \in V_{reg}\}.$$

Computing V^* [Rostalski,Sturmfels].

$$C^T \cdots$$

For non-compact cases, the optimal value c_0^* might be infinte. Also, it might be unattainable.

Introduce *recession cone* at he set of all vectors y.

If K is a convex cone, the the polar $K^0 = \{c \in \mathbf{R}^n | \langle c, x\rangle < 0 \forall x \in K\}$.

Theorem 5 ([GuoWangZhi]) ...

Theorem 6 ([GuoSafeyElDinWangZhi]) *Let $\mathcal{V}^* \subset (P^n)^*$ be the dual variety to the projective closure of V and $C = cl(co(V \cap \mathbf{R}^n))$. If V is irreducibe, smooth and 0^+C is pointed, the V^* is irredcuble and ...*

In unpointed cases, the dimension of $(0^+C)^o$ is strictly less than n.

There are bad parameters values, when ....

3.1.5 Conclusion

We have sown how to compute semi-definite approximations of a noncompant semialgebraic set. Compute the optimal value functions when the feasible region is noncompact.

Given a noncompact convex set C and a convex cone K,

- do there exist an affine subspace $L \subset \mathbf{R}^m$ and a lnear $\pi : \mathbf{R}^m \to \mathbf{R}^n$ ...
- ...

Theorem 7 (Yanniakis) *The minimal m such that C has a $\mathbf{R}_+^m$-lift is equal to the nonnegative rank of its slack matrix.*

See the ISSAC tutorial by Moitra.

Theorem 8 (WangZhi) *Let $C \subset \mathbf{R}^n$ be a polyhedron with at least two vertices. The minimal m such that C has an $\mathbf{R}_+^m$-lift is equal to the nonnegative rank of its extended slack matrix.*

3.2 Software Snippets

3.2.1 Van Ciên Bui etc.

q-defined quasi-shuffle algebras. In Maple. Commands like `Lyndonbasis`.

3.2.2 Tropical Varieties: Yue Ren

For $f = p^a x_1 + p^b x_2 + 1$, the initial term depends on the weightvector, and the fact that this is linear isn't essential. We still decompose (a, b) space into cones, possibly with some tessentaion near the origin.

Tropical geoneters are also interested in the valuation of the coefficients, which we do (SINGULAR) by means of a variable emulating the valuation.

3.2.3 Polynomial Homotopy Continuation on GPUs: Verschelde

This is a SIMT (T=Thread) processing issue. Solution: the reverse mode of automatic differentiation (AD) provides fle-grained paralllelism to evaluate and differentiate all the polynomials in the homotopy, as needed for Newton's mode. Think of the monomial as the unit of data parallelism.

Acceleration compensates for double double and quad double processing (he has libraries). We use K20C, and therefore need to keep 10,000 paths occupied. In double precision (and real double double), we are memory bound. For others we are compute bound.

`phcpy` is the Python scripting interface to PHCpack.

3.2.4 Kat the language of calculations: Mikus Vanags

`function() { return x;}` shows implicit parameter declaration. More sophisticated: `function() { return y~-x;}` or `function() { return y-~x;}` where `~` is the *grace operator*: postfix means move to end of implicit parameter list, prefix means beginning.

Supports both push-button as in old calculator, or drag as in smartphone.

Note similarity with the implicit parameter notations `$0` and `$1` etc. in programming languages

3.3

We generalise [RostalskiSturmfels] to the following.

- When V is nonsmooth and compan, dual varieties of regular locus and sungular locus give Φ for generic γ
- The V is smooth and nocompact, dual valiety

We compute fintelu many paist $(|Phi, Z)$ such that ....

Let $V^* \subset \mathbf{P}^n$ be the dual variety to the projective closure of V and C_h the closuure of the convex hull of $\{V \cap \mathbf{R}^n\}$.

Theorem 9 *If V is equidimensianal and smooth ....*

Write $\Phi = \Phi_0(c_1, \ldots, c_n(c_0^m + \cdots$. Example in which Φ gives no information for some value of γ.

Construct a one-dimensional $C \subseteq V$ such that

$$\sup_{x \in V \cap \mathbf{R}^n} f(x) = \sup_{x \in C \cap \mathbf{R}^n} f(x).$$

. Algorithm [GreuetSafeyElDin2014].

- Construct C
- S_1
- S_2
- S_3

Poalr vareties [many]. A sequence of variaties $\{W_i\}$ wheren W_{n-i+1} is the critical locus of $\ldots$. We need modified polar varieties. $W = \bigcup_{i=1}^{d} M_i$ whnere $M_i = W_{n-i+1} \cap V(X_1, \ldots, X_{i-1})$. Algorithm[Greuetetal] Going through one of S_1, S_2, S_3 get a $\{\Phi_i, Z_i\}$, save this and $Z := Z_i + P_i$ and reconsider $V(Z)$.

γ is bad if either $\Phi_i(c_0, \gamma) = 0$ or $\ldots \not\equiv 0 \wedge \Phi_i(c_0^*, \gamma) \neq 0$.

3.3.1 Singular case

V defined by $x_1^2 + x_2^2 - 1)^3 + 17x_1^2x_2^2$. Then the defining polynomial of V^* is $\Phi_1 := -c_1^2c_2^2 + c_0^2c_1^2 + c_2^2c_0^2$. Let $k - 1, v_k = v$.

1. Compute radical and equidimensional decomposition
2.

This terminates in a finite number of steps.

3.3.2 Conclusion

- How do we compute Φ when the feasible set is neither compact nor smooth.
- How to compute $|Phi$ when the feasible set is a semi-algebraic set.

Q Non-linear objective functions? Where does it break down?

A We haven't considered.

3.4 Probabiloistic Algorothm for Computing the Dimension of Real Algebaric Sets: Bannwarth

Applications: in computationalal real algebaric dimension. Also in engineering, dimension tells us degree of freedom [JinYang2002]. Collins can compute CAD.

Also [Vorobjov,Basu/Pollack/Roy,Koiran]. $O((s+1)\Delta)^{O(d(n-d))}$. Output sensitive, constantinthe exponent is huge, and no lnown pratcical improvements.

We propose a new algorithms for S defined by one polynomial equation. $\widetilde{O}(m^{16}(1+\Delta)^{3d(n-d)+5n+5})$. Probailistic algorithm. Efficient implementation. Uses Gröbner bases rather than geometric resolutions. [HongSafey2012].

1. Random change of variables.
2. Compute boundary$(\pi_i(S))$
3. Compute one point per component
4. lift the fibres.

Let S be defined by $f=0$, and S_ϵ be defined by $f-\epsilon=0$, and is a smooth set.

$$W_i = \overline{\{x \in V | \pi_i(T_\times V) \neq \mathbf{C}^i\}}^{Zar}.$$

[SafeySchost2003]

Theorem 10 *Let S be defined by $f=0$. V_ϵ the algebraic variety of $f-\epsilon=0$. $W\epsilon, i$ be its polar variety, and π_i projections onto $x_1,\ldots,x_i$. Then in generic coordinates ....*

Note the polar variety is

$$f-\epsilon = \frac{\partial f}{\partial x_{i+1}} = \cdots = 0.$$

Some details on avoiding infinitesimals.

1. Compute $\pi_i(V(I_i(f)) \cap \mathbf{R}^n$
2. One point per connencted component. Nearly always the most expensive step — 90+%.
3. Testing fibres $O(n^4(n\Delta^n+n^4)\ldots)$. Expensive when $d=0$.

Complexity is worst when $d \approx n/2$: very much shown in examples. Use FGb and RAGLIB. Compares with CAD. This was nearly always ∞, but Series I/II examples has some decent times, even beating the speakers' for small n. He blamed the linear change of variables.

3.5

Motivation consider tedrahedral die, but unfair. Aticiially asume $p_1+2p_2+3p_3-4p_4=0$. Do 10 trials. So what are the best estimates. This is maximise likelihood function.: maximinse $\prod p_i^{n_1}$, or rather its logarithm. Then use largange multiplies, and the search is for positive solutions. In a paremetric version, the system has three complex solutions, but how often does it have positive real solutions? This is the RRC problem. Use Maple2015.

[HostenKhetanSturmfels2005]. Agan system with unknown p_i (real positive) and λ_i (real,but not necessarily positive) with u_i parameters. For egneric data, there are finitely many complex solutions, the ML-degree.

Standard algoroithm

1. Compute disciminant variety
2. Compute cells determined by variety and number over each cell, e.g. CAD.

Unlike the general case, the discriminant variety of ML-problems are projective varieties. The homogeneous polynomial that generates the reduced co-simension 1 component of the discriminant varioety is calle dthe *data-discriminant* of the Langange likelihood equations. (We can't prove it, but th codimension is in fact always 1)

Example 4 $F = u_0p^2 + u1p + u_0$. $J = 2u_0p + u_1$. *Elimitating* p *gives* $u_1^2 + \cdots$). *But this elimination is expensive.*

Our algorothm.

1. Assumethe output is $D(u + 0, u_1, u_2)$ ssume projection of evaluation is evaluation of projection. Substtute $u_i = r_it + s_i$ where r_i, s_i random. Then compute one-parameter eliminatoin ideadl. Do 100 times and guess total degree. Similarly degrees in each variable.
2. ...

In degre 2, with random models, seems 5 times better than standard algorithm. In degree 3 standard in Macaulay always takes $>$ 2hours, which ors is $\approx$ 100 seocnds: choice of stratgey doens;t seem tomatter much. With literature models, we are much better, but Starategy 2 is much better than definiyely better. Looking at memory pressure, we that Macaulay is using 19.64GB virtual, but ours is 6GB. Real was 8GB.

For 3×3 symmetric model, get D_{XJ} with 1307 terms, total degree 12. This is the locus of multiple values. RAGlib says $D_{XJ} > 0$ has six solutions, $D_{XJ} < 0$ has two. Note that the sign of the discriminant variety is not sufficient information.

3.6 Sparse PolynomialMultiplication: Roche

We would like to multiply any polynomial in time linear in the size of the input, but that's not possible. Sparse polynomials mean the best we can hope for is linear in input+output, but that's not possibly either. Instead softly linear. For dense polynomials, we have naïve. Can do Kronecker encoding, then mutliply (softly linear), and extract.

Sparse is different. Quadratic will always work. Then there's [Yan98], which helps with the overhead. See also [MP14]. In the worst case one can't do better than quadratic, but there are special tecniques, e.g. [vanderHoevenLecerf2012a].

What about sparse interpolation. Evaluate at $T >> \#(fg)$ pints, multiply and interpolate. But Big prime algorithms are too slow (in terms of degree). How about small prime versions? Nontrivial.

By "structural sparsity" we mean where we would have non-zeros irrespective of coefficients. Contrast with "arithmetic sparsity", whch we can't reach.

1. Estimate structural sparsity. Reduce to univariate by a Kronecker substitution (f_K), then make all coefficients 1 (f_s). Compute[1] $\left(f_s^{\bmod p} \cdot g_s^{\bmod p}\right)^{\bmod p'}$, If this is less than half dense, then we'll guess that this is the structural sparsity. If not, double p' and try again.

2. compute structural support: $\left(f_s^{\bmod p} \cdot g_s^{\bmod p}\right)^{\bmod p}$. Check size of structural sparsity. Now encode exponents in the coefficients. Looks like this gets you the product of exponents, but the sum, but $(a\ell+1)\cdot(b\ell+1) = (a+b)\ell \pmod{(\ell^2+1)}$

3. compute arithmetic support (by trimming).

4. compute coefficients. Compute $h_{p,q} = (f_K \cdot g_K)^{\bmod p} \pmod q$ for various q.

Example: 1000 terms, 8 variables 64-bit coefficients and 32-bit coefficients.[2] Computing the structural support is the most expensive step. Note that this is a Monte Carlo algorithm.

A useful routine is sumset. We would like an efficient parallel algorithm! Not yet impemented as sparse interpolation not interpolated. I think this *mght* be practical for *some* cases.

3.7 DKSS: Lüders

De/Kurer/Saha/Saptharishi, based on Fürer's algorithms.

- 64-bit machine with 64-bit word digits. Consider $0 \le a < W^n$ as a vector of n words. C++ with some assembly. `http://www.wrong.com/bignum` under LGPL licence.
- Quadratic
- Karatsuba
- Toom–Cook
- Strassen Let ω be a primitive n-th root of unity. The problem is that, as n grows, we need more and more precision in coefficient ring.

[1]Notation means exponents modulo p.

[2]He had really nice graphics.

- Schönhage–Strassen In $\mathbf{Z}/(2^K+1)\mathbf{Z}$ so $\omega = 2$ is a primitive $2K$th root of unity. $O(N \log N \log\log N)$. This is the standard for $> 150K$ bits.

#words$<$ 28 ordinary; $<$ 152 Karatsuba; $<$ 2464 Toom-Cook with $k = 3$; then S–S.

3.7.1 DKSS

Use $R = P[\alpha]/(\alpha^m + 1)$ with $P = Z/p^c\mathbf{Z}$, where $p = h \cdot 2M + 1$ is prime. Select $M = N/\log^2 N$. Then do a length $2M$ FFT. $2M = -\mu \cdot 2m$. interpret as a $2m \times \mu$ matrix. Do μ legth $2m$ FFTs on the columns using α as root of unity. Then perform "bad multiplications" on the coefficients Then do $2m$ lengthμ FFTs. $O(N \log N K^{\log^* N}$. Since input length is limited by memory, we precompute the primes p (six of them) and generators of $\mathbf{F}_p^*$. Significantly slower than S–S, 28–35 times slower, with the ratio graph being jagged. 80% of the run-time goes onthe bad multiplications. 9% on the pointwise products. Even at this length (28GB), we don't recurse, as the largest coefficients were 195 words (hence TC3 was used here).

Fitting constants, cutover is $N > 10^{10^{4796}}$.

The underlying multiplication is slightly sparse, work maye $\times 1.9$, we could select the parameters more carefully, e.g. 30%. Montgomery reduction might be worth 22%. Even all this would leave DKSS 8.5 times fster.

Chapter 4

9 July 2015

4.1 Algorithms for Finite Field Arithmetic: Schost

$\mathbf{F}_2$, $\mathbf{F}_p$, note that $\mathbf{Z}/4\mathbf{Z}$ is *not* a field. Occur everywhere: number theory, cryptography, coding theory. How do these scale: $\mathbf{F}_5$, $\mathbf{F}_{5^{10}}$, $\mathbf{F}_{5^{100}}$, $\mathbf{F}_{5^{1000}}$, etc. All finite fields are $\mathbf{F}_p[x]/Q(X) = \{a_0 + a_1X + \cdots + a_{d-1}X^{d-1}\}$, but no canonical choice. assume p is given, and let's not talk about normal bases, or Zech logarithms, Conway multiplications.

4.1.1 Basics

Model. $\mathbf{F}_p$ given. In $\mathbf{F}_{p^d}$, represent as a d-vector. Polynomial means $O\left((d \log p)^{O(1)}\right)$. Multiplication, division and XGCD are quasi-linear using FFT, assuming $Q(X)$ given.

For $d = 2$,we want a non-square. $O(1)$ choices randomly, GRH says $(\log p)^{O(1)}$. In higher degreees, need ERH, on [Ivanyosetal]. There is work on special primes[Rónyai, Shoup].

Need cooperation, e.g. if we want $\mathbf{F}_{5^4}$ and $\mathbf{F}_{5^6}$. Note that if $m|n$ then $\mathbf{F}^{p^n} \hookrightarrow \mathbf{F}_{p^m}$. Essentially by polynomial remaindering. However, we don't have a linear-time algorithm here.

Slide from [DeSmitLenstra]. There's a very complete design in MAGMA [Bosma,Cannon,Steel]. See also PARI, NTL, FLINT.

4.1.2 Polynomial Arithmetic

Can write $\mathbf{F}_{11^6}$ either via a polynomial of degree 6, or two of degrees 3 and 2. In the multivariate representation, multiplication is OK, inversion/XGCD more complicated, but complexity is still OK. In general have n variables, and a triangular set of polynomials generating the ideal. But still no quasi-liean algorithms for basic arithmetic.

Change of basis in almost-linear time [Umans,KedlayaUmans,PoteauxSchost] in a boolean model, but this does not seem to be useful yet, as the O-constants build up.

4.1.3 Towers

Motivation: halving on an elliptic curve. Requires extracting two square roots.Hence we build a tower of $\mathbf{F}_{p^{2^k}}$. Similar questions arrive by p-division [Couveignes,DeFeo], hyperelliptic curves [GaudrySchost].

For$p \equiv 1 \pmod 4$, then, if x_0 is not a square, then not only is $x^2 - x_0$, but also $x^4 - x_0$, $x^8 - x_0$ ..., so tower is trivial. For a general $\ell \neq p$, if x_0 is not an ℓth power, $x^{l^i} - x_0$ is irreducible $\forall i$. These are fibres of $x \mapsto x^\ell$.

In general, first go from $\mathbf{F}_p$ to $\mathbf{F}_p(\zeta_\ell)$ then grow $\mathbf{F}_p(\zeta_{\ell^2})$, $\mathbf{F}_p(\zeta_{\ell^3})$, etc. Can reduce $\mathbf{F}_p(\zeta_{\ell^i})$ to $\mathbf{F}_p(x_i)$ witha polynomail Q_i computed by resultants.

Alternatively, use the rule-of-thumb that says that cyclotomic corresponds to elliptic, as in Pollard's $p-1$ and Lenstra ECM. Use isogenies as the equivalent of $x \mapsto x^\ell . \widetilde{O}(\ell^{5+i})$.

Merging $\mathbf{F}_{p^2}$ and $\mathbf{F}_{p_3}$, say. Merged product [BrawleyCarlitz]. If P had roots a_i and Q b_i, let $R = \prod_{i,j}(X - a_i b_j)$. This can be compiuted via a resulatnt [Shoup], or via Newton sums [Bostanetal].

Example 5 $x = \frac{1+\sqrt{5}}{2}$, $y = \sqrt[3]{3}$. *How to know* $x = \frac{1}{6}(xy)^3 + \frac{1}{2}$. *Lnear algebra, an dthis shouldbe turned into ideas with sequences: [Wiedemann], RUR [Rouillier], sparse FGLM [Faugèreetal].*

3-adic towers above $\mathbf{F}_5$: elliptic is better for $i > 10$. Embedding in degree $n(n+1)$. Embed and multmod are both experimentally very close to linear.

4.2 swMATH: Yue Ren

swMATH is an OpenAccess bibliographic database f mathematical software, as a portal for developers and users. Inlcudes informatoin on # publicatoins citing the software. Shows "Top 5 MSC categories for GAP" (by year) as an example of what can be pulled out.

Based on zwMATH, formerly ZentralBlatt. 10K software packages, 90K references out of zwMath. Note that swMath links into zwMath are available even to non ZwMath subscribers.

Impremented at PostgreSQL database, and Django as a Web framework. Currently rank articles by date, but want to improve on this, e.g. actual descriptions before applications etc.

4.3 What's new in Maple 2015

In fact, it's technically 2015.1. 2015.0 was just Maple, whereas 2015.1 includes the new MapleSIM. Large list of areas, so this is merely a slection.

Datasets Some free froma firm called Quantum, who also have a premium access model. Example was oil price (he searched for `OPEC crude`), and he assigned a filtered result to a Maple variable. This integrates with new `dataplot` command, which is largelya new unified interface to a variety of existing options.

Ployhedral Sets and a set of `ExampleSets`. Uses Fourier–Motzkin. Can do the lattice object/faces/edges/vertices/bomm.

IterativeMaps Basically dynamical systems. Goes to compiled code, so fast.

Visualisations Cayley Graphs, Shading between cruves/surfaces.

mathApps Small teaching-oriented worksheets. Built via the new `Explore` command, which has sliders etc.

Now for the meaty stuff.

Statistics e.g. RepeatedMedianEstimator. Also `StudentStatistics`, with a version of `Explore` for random variables. Nice mixture of formulae and displays.

Delay Differential Equations New in 015, via `dsolve`.o

Ordinals JG wrote this. Note that addition etc. are non-commutative.

Modular GCDs Two polynomials. 20 terms. Much faster.

Connectivity A data import command. Understands various formats.

4.4 On the sign of a trigonometric expression: Koseleff

Motivatio: Chebyshev knot diagrams: e.g. $x = T_5(t), y = T_6(t), z = T_7(t)$. In geneal, $z = T_n(t + \phi)$, where ϕ is aphase. Want to know the sign o fthe crosing when projected into plane. Work in $\mathbf{Q}[\cos\frac{\pi}{n}]$. et M_n bethe minimal polynomial of $\cos\frac{\pi}{n}$. Claim that we can decide if $= 0$ on $\widetilde{O}(\ldots)$.

Normally $T_n(x) = \cos n \arccos x$, $U_n(x) = \frac{\sin(n+1)t}{\sin t}$ where $x = \cos t$. We actually use $T_n(x) = 2\cos n \arccos x$. Usual laws. Let $1, T_i$ be our basis for $\mathbf{Z}[x]$. Let $\mathcal{D}$ be the transformation from $\mathbf{Q}[x]$ to be space of even-degree self-reciprocal polynomials: $\mathcal{D}(P) = X^{\deg P} P(x + \frac{1}{x})$. All classical FFT-based complexity results work in new basis. In principle need factorisaton of M_n, but in practice square-free will do.

Chapter 5

10 July 2015: PASCO

5.1 Exploiting Tesla GPUs with MAGMA: Allan Steel

CUBLAS

CUDA native code.

CUSPARSE

CPU Sandybridge 3.1GHz with 384GB memeory, always using one core for comparisons. ATLAS.

GPU C2075 448 CUDE cores, 6GB. Tends to be 15 times faster than ATLAS

GPU' Single K40. 2880 cores. 12GB memory, so 2–3 times faster than previous.

5.1.1 Dense Matrix Multiplication

Use floats/doubles, couple over to GPU and use `CUDA_DGEMM`. Use when $abc \leq 2^{20} \approx 10^9$: this is better than a choice based on a etc. separately. I have a simple CUDA kernel to do the mod p reduction on GPUs. on the GPU.

For (mod 2) multiplication, use 4Russians (4 Ukrainians) method.Take a block of m rows of B at a time, and use a look-uptable. $\lceil\frac{c}{m}\rceil(2^m+r)$ operations on rows of B instead of $\frac{rc}{2}$ classical method. Magma's CPU implementation already uses AVX where available.

I have my own CUDA code, where m is fixed at 8. Avoids bit-fiddling and

- Set Up. Thread (i,j) writes the j-th word of combination i into a buffer $0 \leq i < 256$.
- Apply combinations. Thread (i,j) adds appropriate combinations of works from the buffer.

Choice of block sizes for the grids makes a big difference. Need to avoid writes into global memory (slow).

Then do $\mathbf{F}^{2^k}$ via Karatsuba. When $a = b = c$ the GPU is 3.5–4.9 (depending on 2-ness of a) faster than CPU, GPU' is 10–12. For asymmetric matrices (F_4) get $\times 7$ or $\times 20$.

5.1.2 Dense F_4

My version (MAGMA 2013) of [Fau99]. Look at everything as a dense matrices: no polynomials as such. To compute $T := B - AC^{-1}D$ we have a dense buffer with rows of $[A|B]$. To compute the echelon form, we use the recursive reduction to matrix multiply [Str69]. GPU is a afctor of 3 faster for the *whole* process.

5.1.3 Solving 0-Dim: Evaluation Method

Given a 0-dim $I \subset K[x_1, \ldots, x_n]$, choose $e : 1 \leq e < k$, take an e-tuple $(c_1, \ldots, c_e)$, evaluate here, and compute GB of $I \cup \{x_i = c_i : 1 \leq 1 \leq e\}$, Either we guessed right (then read-off solution, or repeat guessing on next tranche) or get $\{1\}$ (bad guess).

[FS10] use GB techniques to solve minrank. See [Courtois2001] for challenges. A,B easy, C hard. [FS10] estimated 166.7 core-years. MAGMA2012 on a CPU used 1637 seconds for a single instance. (dense F_4 with [JouxVitse2010]).

Using Direct (FGLM) rather than evaluation was about $\times 150$ on a CPU. Also much less sensitive to $\#K$. [FaugereMou2011] used Wiedemann to compute the minimal polynomial of the representation matrix A. Coppersmith's block variation. Much faster in terms of memory access. To recover the vector use [Pau98] version for $K[x]$ of LLL.

So use dense F_4 to get a grevlex basis (very suitable as input dense). MinRank(7,9,4) is a $(1547 + 4811) \times 8K$ problem. [FM13] observe that many rows are singletons, so compress to $b << N$ rows (factor of 5). For Paulus, use Buchberger (dense F_4) for small polynomials. For largr ones, use [Thome2001] Half-GCD-like version of Coppersmith's algorithm.. Maps to fast matrix multiplication over $K[x]$.

Time is dominated by data collection, memory by GB and $K[x]$-LLL.

5.1.4 Results

For $r = 6$ (largest) my solution time (with GPU and 1 core CPU) is $\times 10$ faster forthe GB part.

MinRank c is $(n, k, r) = (11, 9, 8)$. oinors take 1280 seconds (GPUs don't help). 3025 polynomials each of total degree 9. GRlex GB is $\ldots$, A is $N \times n$ with $N = 259545$ and 23.42% dense. 38755 dense rows. Use 800 for Block Wiedemann blocking. (800×38755) by $(\ldots)$ takes 618seconds on CPU, 76.7 withGPU and 28.1 with GPU'. Total times 202h/25.4h/15.1h. 60% Block Wiedemann, 40% dense F_4.

Cyclic 10 takes 3806 seconds on CPU, 2632 on CPU+C2075. Katsura15: CPU+C2075 is worth a factor of 2.

5.2

Working modulo m primes parallelises trivially.

FFT

$$\frac{1}{n}\mathcal{F}^{-1}\left([\mathcal{F}(A,\omega)\odot\mathcal{F}(B,\omega)],\omega^{-1}\right)$$

1. Shuffle
2. Recurse
3. Butterfly

is the standard algorithm, but I found one text giving

1. Butterfly
2. Recurse
3. Shuffle

If we do the forwards operation with (2) and the inverse with (1), the shuffles cancel. This saves memory as well as time.

Also gain 20–25% by better indexing into powers of ω, to take account of recursive calls only using some powers.

CILK parallel implementation. In fact split into four rather than two, which is worth another 20–25%. Doing more doesn't help. Hardcoding the prime (idea due to Montgomery) lets the compiler optimise division, again worth $\times 2$. Can multiply $1024M$ by itself using 2 63-bit primes, with 20 cores given us $\times 16$ over 1 code. Even on one core, we are $\times 2$ over Maple and Magma, which also run out of space earlier.

Q Why not compare with FLINT, etc. Ditto, why not float representations of primes?

A Didn't do FLINT etc. Float's won't represent primes as large.

5.3 Parallel Sparse Interpolation: Roche

[Co-authored with two cadets, written since Jan 2015]

Looking at sample → black box → value. Actually the box is tweakable, e.g. prime/extension field/.... Big prime [**?**]; [Zip79]; Small prime [GrogorievKarpinski].

5.3.1 large prime

1. Choose $q >> \deg f$
2. Find Primitive Roou of Unit ω
3. Evaluate $f(1), f(\omega), \ldots$
4. Berlekamp-Massey
5. Compute roots ζ_i (Lecerf claims improvements here).
6. Compute discrete logs of ζ_i
7. Solve transposed Vandermonde

3,6 are expensive, but parallelise well.

5.3.2 large prime

1. Repeat $O(\log D)$ times steps 2–4
2. Choose $q >> \max c_i$, $p >> T$
3. Evaluate $f(z) \pmod{z^p - 1}$
4. save non-zereo coefficients/expoents
5. Correlate exponents between images
6. solve

Multivariate can be Kronecker, or variable-by-variable [Zip79] or [JavadiMonagan2010a] for parallel-by-variable.

We want to parallelise step 1. Diversification trick [GR11]. So as step 0 we choose random prime q and random $\alpha \in \mathbf{Z}/q\mathbf{Z}$. Heuristically we repeat $\lceil \ell \log D \rceil$ times, and choose $p \approx kT$, where ℓ, k are determined experimentally, hence this is now a heuristic. each process gets p, α, q and blakc box. Returned is a list of triples $(c_? \pmod q), e_? \pmod p), p)$. They come in sorted by primes, and we then resort by coefficients and gather. Need a bound on degree and #terms.

Benchmark from [vanderHoevenLecerf2014]. Our algorithm on 1 core is comparable, and gets a constant speed-up as we throw more cores at it ($\times 4$ on 6 cores).

Still need to improve k, ℓ. Randomised Kronecker [AR14]. Also need to check on better hardware.

5.4 Monagan

Compute $G = GCD(A, B)$ in $\mathbf{Z}[x_1, \ldots, x_n]$. Use many primes (parallel), but this is not enough. Let $\overline{A}$ and $\overline{B}$ be the cofactors, which we will compute. Let $G = \sum_{i=1}^{\deg G} g_i(x_2, \ldots, x_n)x_1^i$ etc., and $t = \max(\#g_i)$. But note the content problem. Note that the univariate GCDs are typically not the time sink.

Instead, consider $G = \sum_{i,j=1}^{\deg G} g_{i,j}(x_3, \ldots, x_n)x_1^i x_2^j$ etc., and $s = \max(\#g_{i,j})$. Content problems are now in *two* variables fewer (reducing to bivariate or univariate, depending on number of variables). Since s is less than t (factor of 3 typically), we win.Inter

Need fast bivariate GCDs. Can interpolate y from univariate images until the GCD stabilises (and check!). But [Bro71] interpolates G and $\overline{A}, \overline{B}$. Stop when $k > \max(\deg_y A, \deg_y B, \deg_y G\overline{A}, \deg_y G\overline{B})$. Nice graph, with an optimisatoin looking for early stabilisation.

Also use FFT with small roots of unity. We evaluate/interpolate in blocks of size j using an FFT of size j ($j = 2, 4, 8, \ldots$). Use Cilk, dense recursive arrays. The number of terms in each was 1.37M, so 10.5MB.

1. Allocate space
2. for $\lceil bnd/j \rceil$ batches in parallel
   (a) Evaluate j images of the inputs into new space in $\mathbf{Z}_p[x_1, x_2]$.
   (b) Make j recursive call in parallel
   (c) ...
3. ...

Space and allocation is quite a problem: as much pre-allocation as possible. 20 cores gets ×10–13, but because were using turbo mode, the maximum is 15.56, not 20. Compared to Maple and MAGMA, even on one core out system is ×10+ better (but these are large dense problems) — even ×1000 over Magma.

5.5 Interatcive Computation and Outsourcing: Roch

Distributed, heterogeneouus, hybrid etc. Various computing levels, also various levels of trust.

What we want is oblivious algorithms: the same for any number of processors. The basic notions are *work* and *deph* — length of critical path.

$$\frac{1}{\Pi_{ave}} \max\left(\frac{W}{p}, D\right) \le Time(p, \Pi) \le \frac{1}{\Pi_{ave}}\left(\frac{W}{p} + D\right)$$

Some schedulers reach $\frac{1}{\Pi_{ave}}\left(\frac{W}{p} + O(D)\right)$. Work-stealing is the obvious implementation. This has interaction between idle and busy processors. If D is small, there is comparatively little stealing: $O(D)$ per processor with high probability. hen both W and D this gives oblivious parallel algorithms.

Example 6 (Parallel prefix) *Given $a_0, \dots, a_n)$, compute $pi_+1, ldots$ whenre $\pi_i = \prod_{k=0}^{i} a_k$. Sequential is efficient, and cache-friendly.*

Example 7 (Frievald's verification) *Verify a matrix product $C = A \cdot B$, where A etc. stored in cloud. Pick a u and check $C.u = A \cdot (B.u)$.*

What about interatcion. Soundness: accept any valid proof. ??: reject (with high probability) an invalid proof.

Example 8 (Graph (non)-isomorphism)

Any problem in PSPACE has a verifier, e.g. #SAT. The key tool is sum-check protocol. Given a Boolean circuit C_n of detph δ that inmplements a function with n input bits. Output= $S_n = \sum_{b_1=0,1} \sum_{b_2=0,1} \cdots \sum_{b_n=0,1} f(b_1, \dots, b_n)$. If $d = 2^\delta$ there are at most d useful gates. Theo: verifier interactively computer

Key 1 Arithmetization

- Transform C_n into C_n^2 over any field K with $\wedge = \times$, $\neg(x) = 1 - x$
- Transform C_n^2 into C_n^K — gates $+$ and $\times$

Key 2 Schwartz–Zippel.

$n = 1$ Computes $S_1 = f_1(1) + f_1(0)$.

Else compute S_n by induction.

- Verifier asks the polynomial $h(y)$ to be prover

$$h(y) = sum_{b_1=0,1} \sum_{b_2=0,1} \cdots \sum_{b_{n-1}=0,1} f(b_1, \dots, b_{n-1}, y)$$

- The verifier recevies $s(y)$ and checks $s(y = h(y)$ by Schwartz–Zippel.

  1.
  2.

This can give us iterated sum certificates. $S = \sum_{i=0,n} f(a_i)$.. The verifier const is the sumcheck protocol, and $O(\log m)$.

5.5.1 Matrix Multiplication

[Thaler Crypto 2003]

- Let A, B be $n \times n$ matrices in K with $m = \log_2 n$.
- A is a boolean functions $\{0,1\}^m \times \{0,1\}^m \to K : A(i_1, \dots, i_m, j_1, \dots, j_m)$ $\dots$

Also interactive error corrcetion.

5.6 Parallel F_4: Pearce

Buchberger's Algorithm with classical division.

- repeated comparisons of the same monomials
- repeated searches for divisors of monomials
- polynomial data structure updates

F_4 either eliminates or amortises these costs.

- Select a batch of syzygies: for $S(f, g)$ put lcm $/LM(f) \cdot f$ and lcm $/LM(g) \cdot g$ in P.
- determine all monomials and reducers
- Sort M and assign indices to monomials
- Perform 9sparse) Gaussian elimination
- new pivots are new leading terms 9add to Gröbner basis)

We wrote a small (30KB) library for $\mathbf{Z}_p; p < 2^{31}$ that does parallel sparse Gaussian elimination in Cilk. We use simple data structures like arrays ("the CPU is pretty smart") and C libraries. Row format is length,index1,coeff1,index2,coeff2,.... The reduction of each row is spawned as a Cilk task. Each thread uses its own buffer to reduce. Rows are added to pivots via compare-and-swap, and may nee to recompute if this fails. For cyclic-9 get $\times 15$ for the GE on 20 cores. Gaussian elimination is 85% of the time, but the overall speedup is only $\times 5$ (and Amdahl's Law says 6.6 is the limit).

Hence need to do better. Spawn the monomial$\times$polynomial as a Cilk task, with a sentinel to protect. In fact, this didn't work out, and we expect we'll need a proper concurrent data structure.

Multi-modular computations. Tried using Maple's Grid package. Again Amdahl burned us over the Chinese Remainder. Good comparisons with Maples FGb and old Magma, but Magma v2.21-4 has radically improved this. Over $\mathbf{Q}$ Maple's FGb is beating us: 101 seconds, while we are 298 on one core, and 105 on 16 cores.

5.7 Gröbner Bases over ANF: Syteenpass

Given $I \subseteq K[x_1, \ldots, x_n]$ wherne $K = \mathbf{Q}(\alpha)$ is a number feild, how to compute a GB of I.

Lett be an extra variable, $f \in \mathbf{Q}[t]$ is the minimal polynomail of α. $S = \mathbf{Q}(\alpha)[X]$, $T = \mathbf{Q}[X, t]$. Fix a flobal product ordering $\succ_K = (\succ_1, \succ_2)$ on $Mon(X, t)$ which is a product ordering. $I \subseteq S$ corresponds to $\tilde{I} \subseteq T$. [Noro2006] noted that lots of $t^k X$ get generates with reducibel t^k get gernated. We use a differnet approach.

Theorem 11 (Chinese Remainder) *R be a Euclidean domain: $m = \prod m_i$ which are coprime*

$$R/\langle m\rangle \equiv R/\langle m_1\rangle \times R/\langle m_2\rangle \cdots R/\langle m_k\rangle$$

So consider $\tilde{I}$ modulo several primes $\tilde{I}_{p_1}$. Admissible–A if f is reducible and square-free (and divides no l.c. or denominator). Suppose $f \pmod{p}_1 = \prod_{i=1}^{r_p} f_i$, and compute Gröbner bases for each f_i, getting $G_{i,1}$ etc. We say that p is admissible of type B of for all $i \neq j$, $\tilde{G}_{i,p}$ and $\tilde{G}_{j,p}$ have the same size and same l.m. set. Then CRT these to give $\tilde{G}_p$. [idreesetal2011a] defined "lucky" Then, as there, we resort to majority voting. Use CRT and rational reconstruction to produce a tentative GB over **Q**. Run a test in positive characteristic to check whether a new prime. Use [Arn03], [Pfister2007] ro check that of $\tilde{I}$ reducte to 0 w.r.t. $\tilde{G}$ and is $\tilde{G}$ is the reduced GV of $\langle\tilde{G}\rangle$, then OK.

With 32 cores, see 3–7× speedup. Generally beat Magma, but not always.

Why does this work? Finite fields, plus lower minimal element degree. And, of course, we're in finite field.

There was a debate other the best way to choose the primes. JHD suggested taking 10 primes, fiding the best cycle shape (which is therefore probably fairly common the the Galois group) and find mor ewith this shape.

5.8

5.9

5.10

Q

A

Chapter 6

11 July 2015

6.1 Numerical Dense Linear Algebra

DRAM performance grows by 23%/year on bandwidth, and Latency 5%. Flops/byte is now well over 10.

Changing chip resolution: at 45nm, a DP FLOP is 100pJ, moving 1m on chip is 6pJ. At 11nm, the FLOP drops to 10pJ, bu tthe moving costs don't change. [JHD: note that we're reaching more transistors in same distance!] [I/O Complexity: the red-blue pebble game: Kung 1981] Loook at citations: peaked in 2013. Was called "I/O", but now it's cache etc., but the same issue.

On a 48-core machine, loooks at Cholesky invesion. MKL LaPack does very poorly, e.g. on 15 threads 30 GFLOPs bersus 110 with perfect scaling from 1 core. ScaLAPACK does better, but not brilliantly.

Paralleise the update: easy and is the $\frac{2}{3}n^3$ term. The factorization is hard to parallelise. So with lookahead we try to hide it, by having all other cores multiplying while one is factoring. In fact, it's the other way round: there's a critical path of factorisatons in whihc the matrix multiplies can be hidden.

Vectorising Original LAPACK idea.

Blocking for L3 BLAS.

Libraries Please tune hard for hardware, and use. See also CUBlas and Alan Steele's talk: section

Autotuning Lot of heuristic searching work. Not my own area.

Variants see FLAME etc. right-looking enable sthe most parallelism, left-looking minimises the number of write I/O, bordered variant is best for schecking whether a matrix is SPD. Note that very variant has its interesting areas.

Recursivity Enables Cache-oblivously system, [Gustavson IBM JRD 1997] [Toldeo SIMAX 1997]

Pipeline [vandeGiejnWatts: SUMMA Concurrency 1997] These use the outer product version of MM algorithms. PUMMA broadcats rows, broadcasts columns, then multiplies, and repeats. Showed a weak scalability example, almost perfect.

2D Block cyclic Distribution The main idea behind ScalaPack. Each processor has blocks from across the matrix, rather than, say processor 1 having just the top left corner, and thus running out of work.

Communcation lower bounds Study ordinary dense MM in the sequential model. Assume two-levels of memory. *Not* allowing Strassen. Square tile times $3.46\left(\frac{n^3}{\sqrt{M}}\right) - M$. Lower bound (Ironyetal2002) $0.35\left(\frac{n^3}{\sqrt{M}}\right)$. $[0.35 = 1/2\sqrt{2}]$. Constraints for maximising number of multiplications

1. total number of reads/writes
2. must fit in cache at start
3. must fit in cache at end

Use Loomis–Whitney inequality. Let $V \subset \mathbf{Z}^3$ be a fniite set, then $|V| \le \sqrt{|V_x||V_y||V_z|}$. Hence

$$2\beta\frac{n^3}{\sqrt{M}} + \frac{2\gamma}{3}n^3,$$

where β, γ are communications and compttaion costs. Looks at Tianhe2 figures Shows thattheir achieved figure is very close to what he expects from this bound (with on-overlapped communication).

Tile Algorithms Very low granularity, scale well. Need a DAG scheduler, block data layout. About five independent developments in 2005–7. One Blue Gene L, on a $1M \times 50$ matrix, get excellent strong scalability with ScaLaPACK. But his ReduceHH (QR3) shows typically 2–3$\times$ the performance.

Weird Work on Orsay/Toulouse/Bordeaux/Sophia, where in practice latency is $c/3$.

Out-of-order scheduling Inversion is in three steps: look at the various DAGs. They can merge very well, . Each has cricial path length $3d + O(1)$, an dthe merged is also $3d + O(1)$, so the merged DAGs are a 3-fold win.

Also run-time schedulers Quark (Tenessee) etc. leads to PLASMA. Also Kaapi from Grenoble, which also understands GPUs.

Heterogeneous I don't do much, only CUBLAS.

6.2 High-Performance Implementation of the Inverse TFT: Johnson

We want t utilise and extend SPIRAL to implement the TFT. We wnat to benefit from SPIRAL's autotuning flow. We used a 45nm Nehalem/Bloomfield, 4 cores, three levels of caches; L3 is 8MB shared at 40 cycles. [vdH04]. Constants matter.

Aim that a hagh-level description should generate everything, e.g. one code of mathematics should produce FFTW. Needs recursion. base cases and target infrastructure. SPIRAL's language SPL, (also Σ-SPL) and OL). Algorithms are rules in SPL and OL. Do a lot of loop merging, vectorisation and parallelisation ourselves, as compilers don't (didn't) seem that good.

Input

$$ModDFT_{n,p,\omega_{rs}} = (ModDFT_{n,p,\omega_r} \otimes I_s)T^n_s(I_R \otimes ModDFT_{n,p,\omega_s})L^n_r$$

is our defintion.

Need [vdH04] do imeplement the TFT when not all terms are being computed.

1. Apply the breakdown rule
2. Convert to Σ-SPL
3. merge loops
4. simplify indices
5. extract required recursion steps

Prime factor algorithms and Rader algorthms already implemented.

Parallel TFT and ITFT can do a few more arithmetic operations to get better vectorisation. Use Montgomery multiplication, and SSE4 doesn't have division. Shows pretty small strircase effects, and practically $\times 2$ as moves fomr 2 to 4 cores. We are as good as FFTW, and have successfully moved SPIRAL to the fixed-point world. Note that Intel's MKL now uses SPIRAL to generate cosine transfroms etc. (in floating)

Note that relaxation, doing some "redundant" arithmetic to improve vectorisation etc. We are trying to produce a "decent" release of SPIRAL.

Q How long does this take?

A A couple of hours.

Q–JG In the floating world, don't you need to worry about stability?

A Using stability as a cost function (for CT algorithms) doens't see to change things.

Q-MM One of the new AVX does $64 \times 64 \mapsto 128$: are you using this?

A Starting to look at.

6.3 3-Ranks for strongly regular graphs: Saunders

Primary motive: Dickson 3^{16} matrix has a 3-rank of 141168 (Dimension 43046721. The applicatoins are to strongly regular graphs. [WengQiuWangXiang2007]. useful for LLL-based recurrence candidate narrowing.

$\mathbf{F}_q = D \cup -D \cup \{0\}$. $D \cap -D = \emptyset$. Laplace Matrix $L_{i,j} = \begin{cases} -1 \\ 0 \\ 1 \end{cases}$

In our case, $D = \{x^2 : x \in \mathbf{F}_q\}$. $q = 3^e$. 3-rank satisfies $x^3 - 4^2 - 2x + 1$ experimentally,and this new result verifies this.

. Gaussian elimination is $\tilde{O}(rn^2)$, whereas we can do $\tilde{O}(n^2 + r^3)$. Space is also $\tilde{O}(r^2)$ rather than $\tilde{O}(n^2)$.

Youse's thesis has a heuristic with certifier. Use 65 compute nodes with 4 Opteron 12cores. RPC and PyRO, and well as OpenMP on nodes.

6.4 Parallel Linear Algebra Dedicated Interface: Sultan

This language, PALADin, is in LinBox. We have POSIX, or pthreads, or Windows threas. Issues of portability, explicit synchronisation etc.

There are annotation-based such as OpenMP and SMPSS. Function class based, such as TBB and Cilk++. xKaapi supports fork-join and dataflow parallel (work stealing).

In the state-of-the-art PLASMA-Quark librray we can see that we need genericity and portability; performance and scalability, and a high level of abstraction. Need to handle unbalanced workloads well. Claim that the answer is alangauge wirth multiple runt-ime systems plugins. We support

- OpenMP 3.x and 4.0 directives, such as OpenMP4 `depend`.
- TBB `parallel_for` etc.
- xKaapi via the libkomp library.

Our implementatoin is maro-based keywords. Tis avoids function-call overheads. Ther is a complementary C++ template meta-programming for loop-cutting techniques.

```
PARFORID (it,0,n,SPLITTER,
    T(it,begin()]=T1[it.begin()]+T2[it.begin()]
```

SPLITTER has a variety of methods:

- `BLOCK_THREADS`
- `BLOCK_FIXED`

- `COLUMN_THREADS`
- `COLUMN_FIXED`
- `ROW_THREADS`
- `ROW_FIXED`
- `GRAIN_SIZE`
- `SINGLE` (i.e. no cutting)

Also various fork/join strategies, such as `PAR_BLOCK`, and dataflow keywords. Five strategies for MATMUL splitting.

TBB does not find the best cutting strategy (160 seconds), whereas we get 30 (`ROW_THREADS`) Again, native OpenMP does $\times 4$ worse than when we specify `ROW_FIXED`. Conversely `ROW_TREADS` is slightly worse than native.

For fork-join, all strategies are good when the matrix is very large, but for smaller ones, it matters. Ditto xKaapi, but OpenMP does not handle the recursion very well.

Our PLUQ does 35% faster than PLASMA-Quark's `dgetrf` ($k = 212$). MKL's `dgetrf` has a disastrous cliff at $N = 11000$ — performance halves.

6.5 PASCO Business Meeting

PG took the chair as CP was on family business. He presented a budget for PASCO 2015.

It was noted that we only had one participant forthe programming challenge. FAS noted that we should do a follow-up exercise with past contestants. The following points were made.

- In the future, AAECC (via Grégoire Lecerf) might well be willing to run a special issue based on PASCO presentations.
- We need a Steering Committee. J-GD, MM, AK and PG volunteered. They were empowered to co-opt any past PASCO chair.
- It would be good to have a more regular schedule, say every other year. The Programming Challenge should also be announced a year in advance, or at least at the first call for papers.

6.6 GPU Accalerated Path Tracking: Verschelde

Polynomial homotopy continuation methods. These are numerical computations, but `double` may not be sufficient. First results in this line in PASCO 2010. f is the system we want to solve, and g is the start system, then consider essentially $tf+(1-t)g$. For quad double, see `http://crd.lbl.gov/~dhbailey/mpdist/qd-2.3.9.tar.gz` [Hidaetal QuadDouble 2001]. Now ported to GPU.

Tracking one part is strictly sequential. So where does parallelism come in? Polynomial evauations and differentiation. See polynomials as a sum of monomials.

1. Common factors and tables of power products
2. Evaluate and differentiation sof produts of variables, via AD, takes $3n-5$ multiplications
3. Coefficient multiplicaton and term summations: order summation jobs by number of terms to balance work.

Path tracker dxoes extrapolate/correct (Newton). See VerscheldeHu (HPCC 2015).

6.7 GPU-acceleration of Optimal Permutatoin-Puzzle Solving: Ishida Naoaki

This is expensive as we are searching in the cayley graph of groups of puzzles. Rubik's cube is 4.33×10^{19}. So we want shortest-path finding: how?

Rubki's diameter is 20 in half-turn metric, and 26 in quarter-turn metric.

Rubik has 8 corner cubies, etc. a state of the cube is 40 variables, 8 corners and 12 edges, 100 bits. However, a better encoding uses 68 bits., for example there are 8! possible curner positions etc. We use a table for state transitions caused by a move.

Iterative Deepening Algorithms: depth-first path search. Have a "pruning and distance" table. Require no repetitions of the same twist, and we fix the order of twists about the same axis.

Toy example has $2^{11} \times 3^7 \times ??? \approx$ 1GB of state table. Can have about 500K states/kernel. Need collaboration between CPU and GPU: CPU does some breadth-first searching until has enough sub-problems to pass to GPU.

For small problems, e.g. 2x2x2 cube, depends drastically on distance function used. Removal of inappropriate moves tends to be worth $\times 10$. Some dependence on search length (on CPU?) and on threads/block, where 64 was the optimal.

For the full Rubik's cube, the CPU time seems to be dominant. We have a tuning method, and are seeing $\times 25$ speedup.

6.8 Sparse Multivariate Polynmial Division: Gastineau

≥ 16 cores.. Distributed format, sorted in some monmial order. School method is repeated cancellation ny leading term. [Wang1996] synchronised all threads between each quotient term. [MP11] parallel sparse using a binary heap. One private heap per thread and a global heap with a lock.

Also various dense methods. Mathemagix is dense, but if the question is sparse this uses much more memory.

We focus on exact division, as in Gauss-Bareiess. [GastineauLaskarCASC2013] Considers an approach with no lock for doing the multiplication (Horowitz pp-matrix). Model is basically single producer, multiple consumers, but the producer can change.

The merge can be done by any fast sequential algorithm: Binary heap [Monagan2009] for the pending part, and a tree with each node has 16 children (4 bits of exponent). Monagan's memory is $\#Q + \#B$, but we are $\#(Q \times B)$, which could be worse.

Practically, Maple is very bad with more than 5–10 (depending) threads, and we degrade slowly. On random polynomials, the tree method is very bad whenthe sparsity is close to 1, but (in general) the heap method were geneally better. We suffer from not having a dynamic switch between quotient heap and remainder heap.. On a large NUMA[1] and PMC, we get typically $\times 100$ for 256 threads.

6.9 Cache oblivious sparse polynomial factoring using the Funnel heap: Abu Salem

[CASC2014]. So how to we exploit delicate data strcutures to speed up algorithms at scale. Also, why does cache oblivious help?

Consider $f \in \mathbf{F}_p[x, y]$. We wnat to consider sufficiently sparse sparse polynomials when $N(f)$ has a few decompositions. We claimthatthe inner workings of Hensel lifting remain oblivious to the sparssity of the input.

Hence we wish to consider a sparse model, distributed.

```
1. for $i := 1 : k - 1$
2.     compute $g_i \cdot h_i$
3.
```

When f is sparse the products are sparse, and the multiplications and additions are memory bound.

So instead we will generalise the priority queue approach. $g_i = \sum g_u^{(i)}$ etc. In [CASC2014] in the worst case scenario, PQ-HL achieves an order of 10 over previous.

Here we implement the priority queue as a Funnel Heap raher than a max heap. Doe sinsert/delete in optimal I/O, and is cahce-oblivious. It reaoganise sitself over updates, and identifies equal monomials (for free). There is a batched mode for chaining.

The maximal element is that in $S_{0,1}$ and A_1. If $S_{0,1}$ is full, we do a SWEEP operation,. Insert and ExtractMax have an amortised cost.

[monaganPeace2009] chained replices outside the binary heap. We don't look for replacements outside $S_{0,1}$.

[1] $5\times$–$8\times$ as expensive for non-local memory.

All monomials α that have to be excluded from the heaps are stored in a static array D.

$$O\left(\frac{kgh}{t'B}+\frac{kg}{\tau}\log\log\frac{kg}{\tau}\right)$$

where τ is the fraction of reduction in the size of the heap during chaining, and τ' is .... $\tau=1$ reverts to the previous scheme.

Large random polynomials over $\mathbf{F}_3$: deg =2,000–20,000, terms up to 1M, but very few (≤ 7) faces in Newton polynomial.

Claim this applies to any Hensel construction, or indeed any sum-of-products construction.

Bibliography

[AR14] A. Arnold and D.S Roche. Multivariate sparse interpolation using randomized Kronecker substitutions. In K. Nabeshima, editor, *Proceedings ISSAC 2014*, pages 35–42, 2014.

[Arn03] E.A. Arnold. Modular algorithms for computing Gröbner bases. *J. Symbolic Comp.*, 35:403–419, 2003.

[BDEW13] R.J. Bradford, J.H. Davenport, M. England, and D.J. Wilson. Optimising Problem Formulation for Cylindrical Algebraic Decomposition. In J. Carette *et al.*, editor, *Proceedings CICM 2013*, pages 19–34, 2013.

[Bro71] W.S. Brown. On Euclid's Algorithm and the Computation of Polynomial Greatest Common Divisors. *J. ACM*, 18:478–504, 1971.

[Dav85] J.H. Davenport. Computer Algebra for Cylindrical Algebraic Decomposition. Technical Report TRITA-NA-8511 NADA KTH Stockholm (Reissued as Bath Computer Science Technical report 88-10), 1985.

[ESY06] A. Eigenwillig, V. Sharma, and C.K. Yap. Almost tight recursion tree bounds for the Descartes method. In *Proceedings ISSAC 2006*, pages 71–78, 2006.

[Fau99] J.-C. Faugère. A new efficient algorithm for computing Gröbner bases (F_4). *J. Pure Appl. Algebra*, 139:61–88, 1999.

[FM13] J.-C. Faugère and C. Mou. Sparse FGLM algorithms. `http://arxiv.org/abs/1304.1238`, 2013.

[FSEDS10] J-C. Faugère, M. Safey El Din, and P.-J. Spaenlehauer. Computing Loci of Rank Defects of Linear Matrices using Gröbner Bases & Applications to Cryptology. In S.M. Watt, editor, *Proceedings ISSAC 2010*, pages 257–264, 2010.

[GR11] M. Giesbrecht and D.S. Roche. Diversification improves interpolation. `http://arxiv.org/abs/1101.3682`, 2011.

[JdM12] D. Jovanović and L. de Moura. Solving Non-Linear Arithmetic. In *Proceedings IJCAR 2012*, pages 339–354, 2012.

[KK05] E. Kaltofen and P. Koiran. On the complexity of factoring bivariate supersparse (lacunary) polynomials. In *Proceedings ISSAC 2005*, pages 208–215, 2005.

[Len99a] H.W. Lenstra Jr. Finding small degree factors of lacunary polynomials. *Number theory in progress*, pages 267–276, 1999.

[Len99b] H.W. Lenstra Jr. On the factorization of lacunary polynomials. *Number theory in progress*, pages 277–291, 1999.

[MP11] M. Monagan and R. Pearce. Sparse polynomial division using a heap. *J. Symbolic Comp.*, 46:807–822, 2011.

[MP14] M. Monagan and R. Pearce. The design of Maple's sum-of-products and POLY data structures for representing mathematical objects. *ACM Comm. Computer Algebra*, 48:166–186, 2014.

[Pan00] V.Y. Pan. Matrix Structure, Polynomial Arithmetic and Erasure-resilient Encoding/Decoding. In C. Traverso, editor, *Proceedings ISSAC 2000*, pages 266–271, 2000.

[Pau98] S. Paulus. Algorithms for CM-fields. In J. Buhler, editor, *Proceedings 3rd Algorithmic Number Theory Symposium*, pages 567–575, 1998.

[Str69] V. Strassen. Gaussian Elimination is not Optimal. *Numer. Math.*, 13:354–356, 1969.

[SY12] V. Sharma and C.K. Yap. Near Optimal Tree Size Bounds on a Simple Real Root Isolation Algorithm. In *Proceedings ISSAC 2012*, pages 319–326, 2012.

[vdH04] J. van der Hoeven. The Truncated Fourier Transform and Applications. In J. Gutierrez, editor, *Proceedings ISSAC 2004*, pages 290–296, 2004.

[Yan98] T. Yan. The geobucket data structure for polynomials. *J. Symbolic Comp.*, 25:285–294, 1998.

[Zip79] R.E. Zippel. Probabilistic Algorithms for Sparse Polynomials. In *Proceedings EUROSAM 79*, pages 216–226, 1979.