Using the distribution of cells by dimension in a cylindrical algebraic decomposition

James Davenport (Bath)

With Russell Bradford, Matthew England and David Wilson (Bath)

22 September 2014

Background

Cylindrical Algebraic Decomposition (CAD) is a powerful tool in Real Algebraic Geometry

- Quantifier Elimination (original use [Col75])
- Robot Motion Planning ([SS83, WBDE14a])
- Branch cuts/formula simplification ([BD02])

But the complexity is large in theory (doubly exponential in the number of variables [BD07]), and in practice (multi-hour runs/heap exhausted)

How to compute a CAD

These days there are fundamentally two approaches

P/L (Projection and Lifting) [Col75, and many improvements]

- finer projection operators [McC88];
- Partial CAD to make use of the quantified structure of a formula when lifting [CH91];
- the use of equational constraints [McC99];
- truth-table-invariant CADs to apply equational constraint techniques more widely [BDE+13];

RC (Regular Chains) an alternative approach where the problem is solved in complex space and then refined to a CAD of real space [CMMXY09, CMM12].

This paper is about P/L, though we ask about RC at the end.

P/L (initial) Method: outline

$\mathcal{P}_n \to \mathcal{P}_{n-1}$	Project on $n-1$ variables
$\vdots$	and so on
$\mathcal{P}_2 \to \mathcal{P}_1$	Project to univariate polynomials
Isolate	All the roots of $\mathcal{P}_1$, which is a c.a.d. of $\mathbf{R}^1$
Lift	To a c.a.d. of $\mathbf{R}^2$ sign-invariant for $\mathcal{P}_2$
$\vdots$	and so on
Lift	To a c.a.d. of $\mathbf{R}^n$ sign-invariant for $\mathcal{P}_n$

In practice, lifting is by far the most expensive part: the full-dimensional cells have rational sample points, but the lower-dimensional ones, in general, have algebraic number coordinates, leading to the manipulation of roots of polynomials with algebraic number coefficients (ouch!).

Hence one can consider

Just the cells of full dimension [McC93, Str00, both in a P/L context].

By $\langle f(y), n_1, n_2 \rangle$ we mean that (unique) root of $f(y) = 0$ lying between n_1 and n_2, $n_i \in \mathbf{Q}$ (in practice $\in \mathbf{Z}[1/2]$)

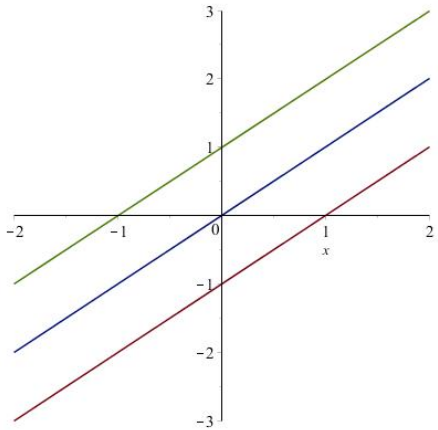
Then a cell of full dimension is

- x_1 between $\langle f_1(x_1), n_1, n_1' \rangle$ and $\langle f_2(x_1), n_2, n_2' \rangle$
- Let n_3 be a (the simplest) number in $[n_1', n_2]$
- x_2 between that branch of $g_1(x_1, x_2)$ going through $\langle g_1(n_3, x_2), n_4, n_4' \rangle$ and that branch of $g_2(x_1, x_2)$ going through $\langle g_2(n_3, x_2), n_5, n_5' \rangle$
- Let n_6 be a (the simplest) number in $[n_4', n_5]$
- x_3 between that branch of $h_1(x_1, x_2, x_3)$ going through $\langle h_1(n_3, n_6, x_3), n_7, n_7' \rangle$ and that branch of $h_2(x_1, x_2, x_3)$ going through $\langle h_2(n_3, n_6, x_3), n_8, n_8' \rangle$
- $\cdots$ and so on

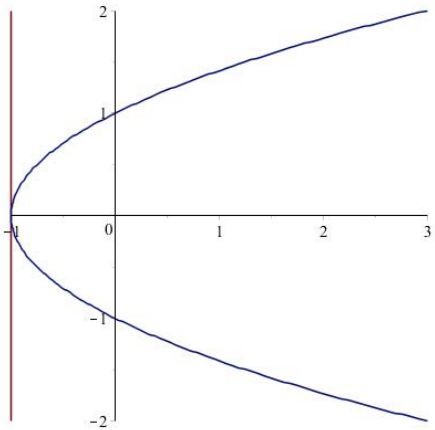
All described in terms of rational numbers and roots in $\mathbf{Q}[x]$.

How might numbers of cells be correlated between full-dimensional cells and all cells?

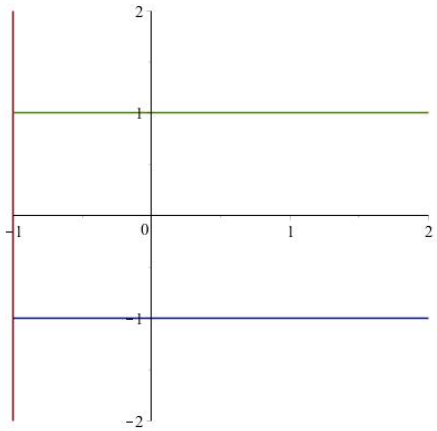
Not perfectly (though Euler's formula applies): four 2-dimensional cells in $\mathbf{R}^2$ might come from any of



4×2D;3×1D,0×0D



4×2D;4×1D,1×0D

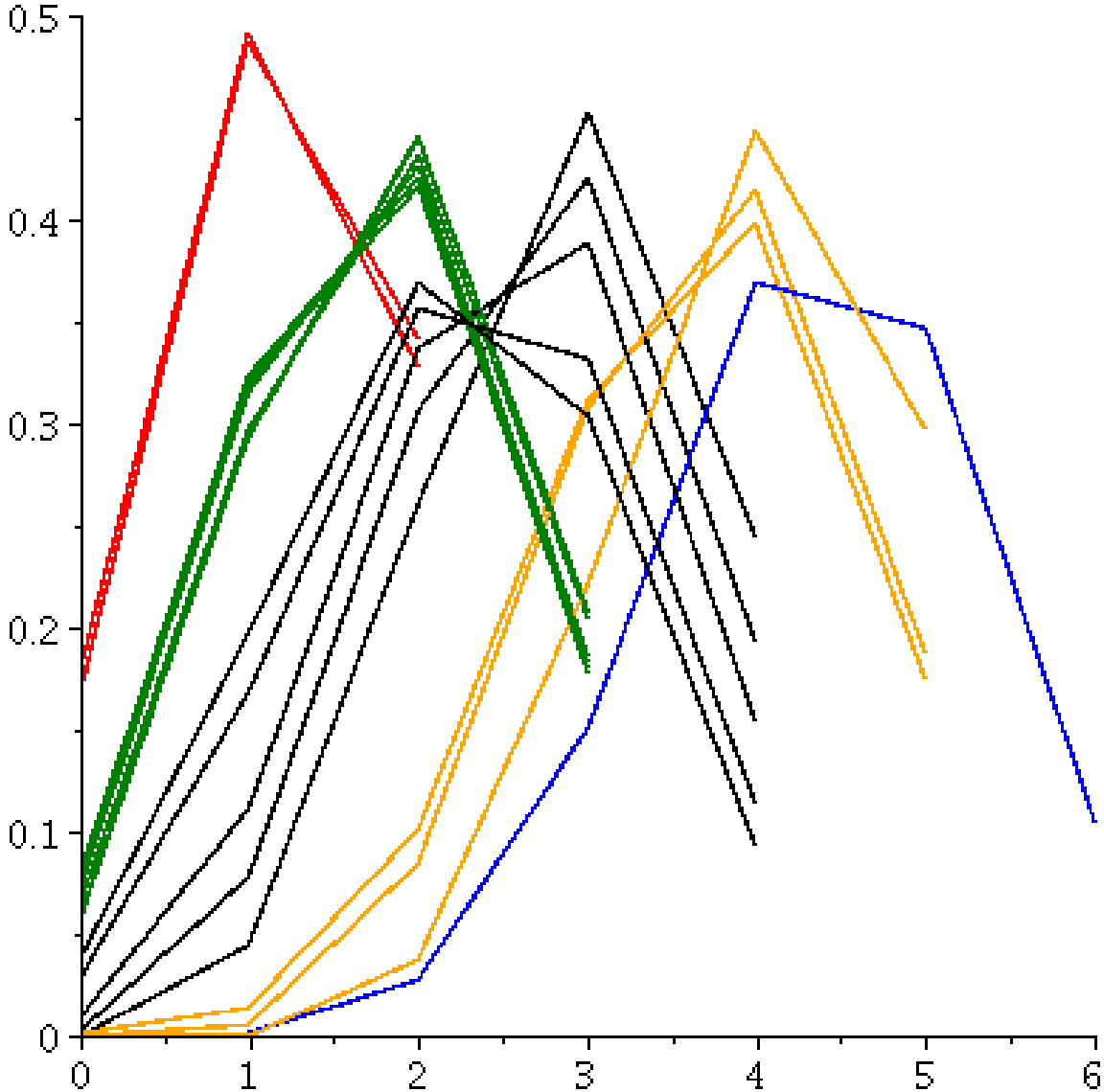


4×2D;5×1D,2×0D

Though no known algorithm will construct it, the last could also be 4/4/1 or even 4/3/0: such CADs are not *well-bordered*

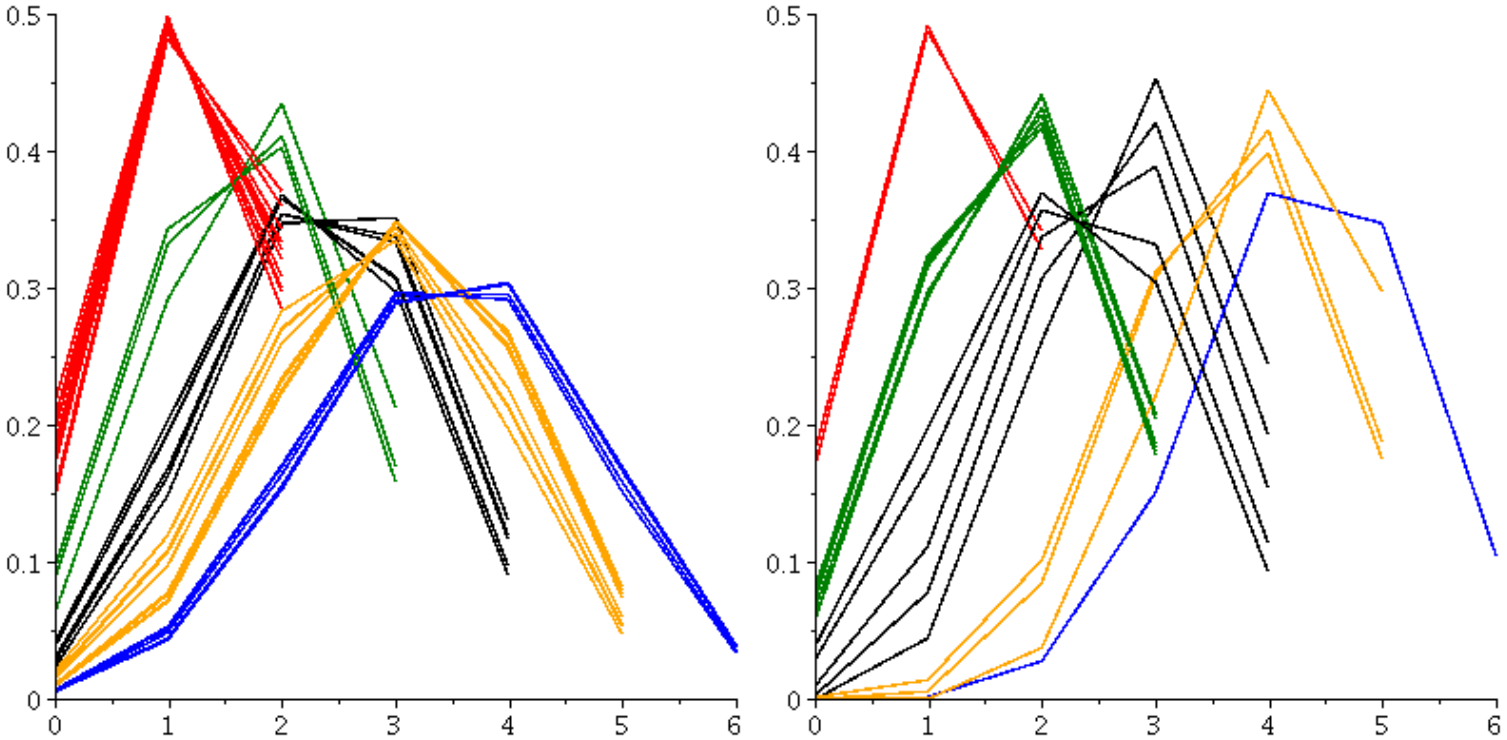
Experimental Evidence (I: Real CADs)

Figure: CAD cell dimension distribution for examples in [WBD12]. Lines coloured the same relate to problems with the same number of variables .



Experimental Evidence (II: Random Constructions)

Figure: Comparing the distribution of cell dimensions of combinatorially random CADs with those created for real examples.



Combinatorially random

Examples from [WBD12]

If I'm interested in the whole CAD why would I want just the full-dimensional cells?

Because CAD actually has a lot of choices, and knowing (cheaply) the number of full-dimensional cells might help us make the right choice.

- Choice of variable ordering (where allowed)
- Choice of *designated* equational constraint (TTICAD)
- Choice of formulation (negation etc.) [BDEW13]

These choices can easily mean a factor of $\times 10$ in running time, and sometimes spell the difference between feasibility and infeasibility. Also, for a given problem, #cells is strongly correlated with overall running time.

Furthermore it is possible to construct

Just the cells of full dimension [WBDE14b, in a P/L context [McC85]]: a 1-layered CAD
in such a way that the cells of lower dimension can be "infilled" later from the information already obtained (*lazy* CAD).
The theoretical complexity of this is essentially the fourth root of the complexity of the base [McC85] process, and in practice [WBDE14b] a factor of 20–40 times faster for four-variable problems.
Hence we could compute the number of full-dimensional cells for various options, take one with fewest cells, and refine this to a complete decomposition.
This heuristic looks at the *complete* real geometry, unlike the `ndrr` heuristic of [BDEW13] which just looked at the geometry of $\mathbf{R}^1$.

Example I --- Three variables

Table: Using 1-layered sub-CADs as a heuristic to pick the variable ordering for a CAD for $\left\{x^2+y^2+z^2-1, xy-yz+3, x+y-yz^4\right\}$.

		Cells		Time	
Order	1-LCAD	Prediction	CAD	1-LCAD	CAD
$x \succ y \succ z$	**118**	**615**	**539**	**0.341**	**2.919**
$x \succ z \succ y$	160	833	789	0.474	5.720
$y \succ x \succ z$	340	1771	1799	0.736	13.055
$y \succ z \succ x$	432	2250	2211	0.950	30.638
$z \succ x \succ y$	224	1167	1133	0.549	10.156
$z \succ y \succ x$	392	2042	2117	0.779	50.150

The total time for all six 1-layered sub-CADs is 3.829 seconds which, along with computing the CAD for $x \succ y \succ z$, means that it would take 6.748 seconds to obtain a CAD using this heuristic. Maximum saving of 43.402 seconds over computing directly the CAD for $z \succ y \succ x$ and 1672 cells over the CAD for $y \succ z \succ x$. On average our CAD would have taken 18.773 seconds/ 1431 cells. Average savings 892 cells (62%) and 12.025 seconds (64%).

Example II --- Four variables

Table: Using 1-layered sub-CADs as a heuristic to pick the variable ordering for $\{a^2+b^2+c^2+d^2-1, a^2-4, a-d, b-c, a-c, b-1\}$.

Sample Order	Cells 1-LCAD	Cells Prediction	Cells CAD	Time 1-LCAD	Time CAD
$a \succ b \succ c \succ d$	7640	47453	75923	10.242	387.233
$a \succ c \succ d \succ b$	11196	69540	108753	13.945	557.628
$b \succ c \succ d \succ a$	1224	7602	10933	**1.676**	52.973
$d \succ b \succ c \succ a$	**1172**	**7280**	**10213**	2.989	**51.696**
20 more	⋮	⋮	⋮	⋮	⋮

The 1-layered (full dimensional) cell count correctly identifies the ordering with the most efficient complete CAD. Timings would have identified another (nearly as good). The total time to compute all 1-layered sub-CADs is 132.238 and so using the heuristic to produce a CAD takes 183.934 seconds. On average the CAD would have 27921 cells and take 194.382 seconds. Hence for this example the heuristic would save a significant number of cells but the **net** time savings would be modest due to 24× 1-layer.

Example II --- Some Statistics

Cells all versus Cells 1 10.10*x-1832; r^2=0.9985
Time all versus Time 1 40.74*x-30.23; r^2=0.9688
Time all versus Cells 1 0.05245*x-14.97; r^2=0.9951
Time all versus Cells all 0.005242*x-5.512; r^2=0.9979
Time 1 versus Cells 1 0.001246*x+0.5359; r^2=0.9635

Example III --- Four variables/ 75 examples

Table: Using the number of full dimensional cells to pick the variable ordering for 75 random examples.

	Cells		Time	
	Problem Max	Problem Av	Problem Max	Problem Av
Average	$4,719$	$2,220$	64.3	10.0
Example	55.0%	38.7%	38.7%	12.9%
Best	$12,816$	$5,204$	631.9	143.3
Example	93.6%	84.6%	84.6%	70.1%
Worst	762	297	-44.1	-66.1
Example	13.9%	6.49%	-27.9%	-49.3%

We found that on average the heuristic saved 38.7% of the cells and 12.9% of the computation time for a problem when compared to picking an ordering at random. However, this masks a lot of variance in the data (see paper).

Further Developments

We can use the same technique for other formulation questions.

- Which equational constraint? The methodology in [BDE+13] privileges one equational constraint. We show a small example in the paper, using this to decide which.
- For the "ladder in corridor" problem (SYNASC 2013) we lifted, not the full-dimensional cells, but those of full *relative* dimension on the variety $(x - x')^2 + (y - y')^2 = 9$. This took 200 seconds for the [WBDE14a] formulation, but timed out for [Dav86] formulation. This accords with the full CAD (5 hours/timed out on QEPCAD).

Work in Progress

We have family of heuristics for PL-CAD of increasing computational cost

1. Brown's heuristic (very cheap) [Bro04]
2. `sotd` [DSS04]
3. `ndrr` [BDEW13]
4. 1-layer (this paper)

The first three have been compared [HEW+14a], and a machine-learning blend was found to be better (in terms of cell count) than any one [HEW+14b]. This should be extended to this heuristic.
There are also heuristics for RC-CAD (from RC code; [EBDW14]), and again we should compare.

Future work: Parallelism

- Running all possible choices is pathological parallel.
- Refining from a 1-layer CAD to a full CAD is also very parallel.
-  Different cells are not guaranteed to have the same algebraics, e.g. $2\sqrt{2}$ versus $\sqrt{8}$.
- Also allows "early abort".
- In example 4, going with the first to finish would cost $24 \times 1.676 + 52.973 = 93.2$ seconds.
- Going with the smallest cell count would cost $24 \times 2.989 + 51.696 = 123.4$ seconds, plus wasted time going with first to finish.
- Compared with average 194.4 seconds,
- or 183.9 seconds if we run all 24 1-layer CADs to conclusion.

Definitely worth investigating.

Future work: Regular Chains

This is all based on the PL-approach and computing layered CADs [WBDE14b].
The Regular Chains approach first computes a cylindrical decomposition of $\mathbf{C}^n$, then (`MakeSemiAlgebraic` operation) converts it to $\mathbf{R}^n$.
Though that's not the way the code is structured (nor was our PL-CAD to begin with!), it should be possible to count just the full-dimensional cells more cheaply than computing everything

Bibliography


R.J. Bradford and J.H. Davenport.
Towards Better Simplification of Elementary Functions.
In T. Mora, editor, *Proceedings ISSAC 2002*, pages 15–22, 2002.


C.W. Brown and J.H. Davenport.
The Complexity of Quantifier Elimination and Cylindrical Algebraic Decomposition.
In C.W. Brown, editor, *Proceedings ISSAC 2007*, pages 54–60, 2007.


R.J. Bradford, J.H. Davenport, M. England, S. McCallum, and D.J. Wilson.
Cylindrical Algebraic Decompositions for Boolean Combinations.
In *Proceedings ISSAC 2013*, pages 125–132, 2013.

Bibliography

II


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.


C.W. Brown.
Tutorial handout.
`http://www.cs.usna.edu/~wcbrown/research/ISSAC04/handout.pdf`, 2004.


G.E. Collins and H. Hong.
Partial Cylindrical Algebraic Decomposition for Quantifier Elimination.
J. Symbolic Comp., 12:299–328, 1991.

Bibliography

III


C. Chen and M. Moreno Maza.
An Incremental Algorithm for Computing Cylindrical Algebraic Decompositions.
http://arxiv.org/abs/1210.5543, 2012.


C. Chen, M. Moreno Maza, B. Xia, and L. Yang.
Computing Cylindrical Algebraic Decomposition via Triangular Decomposition.
In J. May, editor, *Proceedings ISSAC 2009*, pages 95–102, 2009.


G.E. Collins.
Quantifier Elimination for Real Closed Fields by Cylindrical Algebraic Decomposition.
In *Proceedings 2nd. GI Conference Automata Theory & Formal Languages*, pages 134–183, 1975.

Bibliography

J.H. Davenport.
On a "Piano Movers" Problem.
SIGSAM Bulletin 1/2, 20:15–17, 1986.

A. Dolzmann, A. Seidl, and Th. Sturm.
Efficient Projection Orders for CAD.
In J. Gutierrez, editor, *Proceedings ISSAC 2004*, pages 111–118, 2004.

M. England, R. Bradford, J.H. Davenport, and D.J. Wilson.
Choosing a Variable Ordering for Truth-Table Invariant Cylindrical Algebraic Decomposition by Incremental Triangular Decomposition.
In *Proceedings ICMS 2014*, pages 450–457, 2014.

Bibliography

V

Z. Huang, M. England, D. Wilson, J.H. Davenport, and L.C. Paulson.
A comparison of three heuristics to choose the variable ordering for CAD.
http://arxiv.org/abs/1405.6082, 2014.

Z. Huang, M. England, D. Wilson, J.H. Davenport, L.C. Paulson, and J. Bridge.
Applying machine learning to the problem of choosing a heuristic to select the variable ordering for cylindrical algebraic decomposition.
In S.M.Watt *et al.*, editor, *Proceedings CICM 2014*, pages 92–107, 2014.

Bibliography

S. McCallum.
An Improved Projection Operation for Cylindrical Algebraic Decomposition.
Technical Report 548 Computer Science University Wisconsin at Madison, 1985.

S. McCallum.
An Improved Projection Operation for Cylindrical Algebraic Decomposition of Three-dimensional Space.
J. Symbolic Comp., 5:141–161, 1988.

S. McCallum.
Solving polynomial strict inequalities using cylindrical algebraic decomposition.
Computer J., 36:432–438, 1993.

Bibliography

S. McCallum.
On Projection in CAD-Based Quantifier Elimination with Equational Constraints.
In S. Dooley, editor, *Proceedings ISSAC '99*, pages 145–149, 1999.

J.T. Schwartz and M. Sharir.
On the "Piano-Movers" Problem: II. General Techniques for Computing Topological Properties of Real Algebraic Manifolds.

Adv. Appl. Math., 4:298–351, 1983.

A.W. Strzeboński.
Solving systems of strict polynomial inequalities.
J. Symbolic Comp., 29:471–480, 2000.

Bibliography


D.J. Wilson, R.J. Bradford, and J.H. Davenport.
A Repository for CAD Examples.
ACM Communications in Computer Algebra 3, 46:67–69, 2012.


D.J. Wilson, R.J. Bradford, J.H. Davenport, and M. England.
A "Piano Movers" Problem Reformulated.
In *Proceedings SYNASC 2013*, pages 53–60, 2014.


D.J. Wilson, R.J. Bradford, J.H. Davenport, and M. England.
Cylindrical Algebraic Sub-Decompositions.
Mathematics in Computer Science, 8:263–288, 2014.