Generating Subfields

Mark van Hoeij

June 15, 2017

Overview

Papers:

1. *Generating Subfields* (vH, Klüners, Novocin) ISSAC'2011.
2. *The Complexity of Computing all Subfields of an Algebraic Number Field* (Szutkoski, vH), Submitted to JSC.
3. *Functional Decomposition using Principal Subfields* (Allem, Capaverde, vH, Szutkoski) ISSAC'2017.

Implementations:

- (1): Nicole Sutherland, in Magma.
- (2),(3): Jonas Szutkoski, www.math.fsu.edu/∼jszutkos

  Planning to add to Magma.

Applications of subfields

Example 1. Use a CAS to solve this system of equations:

$$a^2 - 2ab + b^2 - 8 = 0, \quad a^2b^2 - (a^2 + 2a + 5)b + a^3 - 3a + 3 = 0$$

Result: $a = \alpha, \quad b =$

$$\frac{-17\alpha^7}{1809} + \frac{61\alpha^6}{3618} + \frac{371\alpha^5}{1809} - \frac{1757\alpha^4}{3618} - \frac{563\alpha^3}{603} + \frac{6013\alpha^2}{3618} + \frac{3184\alpha}{1809} + \frac{7175}{3618}$$

where α denotes a root of

$$x^8 - 20x^6 + 16x^5 + 98x^4 + 32x^3 - 12x^2 - 208x - 191 = 0.$$

Example 1 has a simpler solution:

$$a = \sqrt{3} + \sqrt[4]{2} - \sqrt{2}, \quad b = \sqrt{3} + \sqrt[4]{2} + \sqrt{2} \tag{1}$$

To find it we first need subfields of $\mathbb{Q}(\alpha)$.

Applications of subfields

Bostan and Kauers [Proc AMS 2010] gave an algebraic expression for the generating function for Gessel walks, using two minpoly's with a combined size of 172 Kb. By computing subfields, this expression could be reduced to just 300 bytes, a 99.8% reduction.

Why did computing subfields reduce the expression size?

When $\mathrm{char}(k) = 0$, then a tower of algebraic extensions

$$k \subseteq k(\alpha_1) \subseteq k(\alpha_2) \subseteq k(\alpha_3) = K$$

can be given by a single extension $K = k(\alpha)$.

The **primitive element theorem** produces such α with a minpoly that is usually large.
So we can expect the reverse process (computing subfields) to reduce expression sizes.

Notations.

Let $K = k(\alpha)$ be a separable field extension of k of degree n with minpoly f.

Goal: Find all subfields of K/k, hopefully efficient in practice as well as in theory.

Theoretical issue: There is no polynomial time algorithm because there could be more than polynomially many subfields.

Can compute in polynomial time: a **generating set** $\{L_1, \ldots, L_r\}$

$$\{\text{subfields of } K/k\} = \{\text{intersections of } L_1, \ldots, L_r\}$$

The Subfield Polynomial

Let $k \subseteq k(\alpha) = K$ be an algebraic extension with minpoly f. Let $k \subseteq L \subseteq K$ be a subfield. Let $g \in L[x]$ be the minpoly of α over L.

Definition: We call this g the **subfield polynomial** of L.

$g \rightsquigarrow L$ (can find L from g)

To be precise: L is generated by the coefficients of g.

Note: A subfield polynomial is a factor of f in $K[x]$.

So we could find all subfields by trying every factor of f in $K[x]$.

Factors of f

Let $f = f_1 \cdots f_r$ be the factorization of f in $K[x]$. We can assume that $f_1 = x - \alpha$.

Finding Subfields, Exponential Complexity:
For each of the 2^r monic factors of f in $K[x]$, compute the field generated by the coefficients of that factor.

Finding Subfields, Polynomial Complexity:
Perform a computation for each polynomial $f_2, f_3, \ldots, f_r$.

Problems:

1. These $f_2, f_3, \ldots$ are not subfield-polynomials;
   their coefficients do not lead to proper subfields.
2. And even if they did, we wouldn't get every subfield.

Finding subfields

Let $f = f_1 \cdots f_r$ be the factorization of f in $K[x]$, with $f_1 = x - \alpha$.

Define the i'th **principal subfield**

$$L_i = \{h(\alpha) \mid h(x) \in k[x]_{<n} \text{ and } h(x) \equiv h(\alpha) \bmod f_i\}.$$

The condition

$$h(x) \equiv h(\alpha) \bmod f_i$$

translates into k-linear equations for the coefficients of h.
So

$$h(\alpha) \in L_i \iff \text{linear equations for coeffs}(h).$$

A generating set

A set S of subfields of K/k is a **generating set** if every subfield of K/k is an intersection of members of S.

Theorem: The **principal subfields** $L_2, \ldots, L_r$ from the previous slide form a **generating set**.

Theorem: If $k = \mathbb{Q}$ then a generating set can be computed in polynomial time.

After that we find all subfields by computing intersections. The cost depends linearly on m, the number of subfields.

(m can be more than polynomial in n).

Finding all subfields

Phase 1: Find a generating set.

Phase 2: Compute intersections to find all subfields.

Notation: m is the number of subfields.

Practical performance: Phase 1 usually dominates the CPU time unless m is large.

Theoretical complexity: Phase 2 dominates the theoretical complexity because Phase 1 is polynomial time, but m is not polynomially bounded.

Finding all subfields

Phase 1: Find a generating set.

Phase 2: Compute intersections to find all subfields.

ISSAC'2011 introduced "principal subfields" / "generating set" and algorithms to compute them.

To optimize theoretical complexity one needs optimize Phase 2. This was done in recent joint work with Jonas Szutkoski.

Result: better complexity, and better CPU times if m is large.

Tricky part: Do not want to be slower for small m.

$\rightsquigarrow$ The data used to speed up Phase 2 must be computed quickly.

Fast intersections

ISSAC'2011:

- Each subfield L of K/k is a k-vector space. So any two subfields L_1, L_2 can be intersected with k-linear algebra.
- So after Phase 1 (computing principal subfields) all other subfields can be computed with **linear algebra**.
- If m is large, then there are many **intersections** to compute.

New idea: Represent a subfield L with some data P_L such that:

1. $L \rightsquigarrow P_L$ is fast (for principal subfields)
2. P_L is small (for any L)
3. $(P_{L_1}, P_{L_2}) \rightsquigarrow P_{L_1 \bigcap L_2}$ is fast. (for any L_1, L_2)
4. $P_L \rightsquigarrow L$ is fast (for any L)

Fast intersections: Use (3) instead of linear algebra.

Fast intersections, first try

Factor $f = f_1 \cdots f_r \in K[x]$ where $K = k(\alpha)$.
If L is a subfield of K/k, then its *subfield polynomial* $g \in L[x]$ (the minpoly of α over L) is a factor of f. So

$$g = \prod_{i \in S_L} f_i \quad \text{for some} \quad S_L \subseteq \{1, \ldots, r\}.$$

S_L encodes the subfield polynomial.
Does that meet the requirements?

1. $L \rightsquigarrow S_L$ is fast (nontrivial)
2. S_L is small (definitely!) (only r bits)
3. $(S_{L_1}, S_{L_2}) \rightsquigarrow S_{L_1 \bigcap L_2}$ is fast (not enough data in S_{L_1}, S_{L_2})
4. $S_L \rightsquigarrow L$ is fast ($S_L \rightsquigarrow g \rightsquigarrow$ generators of L)

To intersect quickly, we need slightly more data than S_L.

Fast intersections, second try

Factor $f = f_1 \cdots f_r \in K[x]$ where $K = k(\alpha)$.

If L is a subfield of K/k, then the factorization of f over L defines a *partition* P_L of $\{1, \ldots, r\}$.

Here i, j are in the same part if f_i, f_j divide the same irreducible factor of f in $L[x]$.

P_L encodes the factorization of f over L. Meets requirements?

1. $L \rightsquigarrow P_L$ is fast (nontrivial)
2. P_L is small (only $r \cdot \log(r)$ bits)
3. $(P_{L_1}, P_{L_2}) \rightsquigarrow P_{L_1 \bigcap L_2}$ is fast
4. $P_L \rightsquigarrow L$ is fast

After computing P_L for each generating subfield, the entire subfield lattice can be found quickly (item 3) where each subfield is represented in a convenient way (items 2 and 4).

Partition P_L example

Let $K = k(\alpha)$, minpoly $f \in k[x]$, and factor $f = f_1 \cdots f_r \in K[x]$.
May assume $f_1 = x - \alpha$.

Let L be a subfield of K/k. The factorization of f in $L[x]$ is:
$f = g_1 \cdots g_d$ for some $1 \leq d \leq r$.
Since $L \subseteq K$, each g_i is a product of some f_j's.

Example:
Suppose $r = 9$ and $g_1 = f_1 f_2 f_9$, $g_2 = f_3 f_4$, and $g_3 = f_5 f_6 f_7 f_8$.
Then the partition P_L is:

$$P_L = \{\{1, 2, 9\}, \{3, 4\}, \{5, 6, 7, 8\}\}$$

$P_L \rightsquigarrow$ "the part with 1" $= \{1, 2, 9\} \rightsquigarrow f_1 f_2 f_9 = g_1 \rightsquigarrow L$

because $g_1 =$ subfield polynomial, so $L = k(\text{coeffs}(g_1))$.

Partition P_L example (diagram)

$$\begin{array}{lll}
K & f = f_1 f_2 f_3 f_4 f_5 f_6 f_7 f_8 f_9 & \longrightarrow P_K = \{\{1\},\{2\},\{3\},\{4\},\{5\},\{6\},\{7\},\{8\},\{9\}\} \\
| & \downarrow & \downarrow \\
L & f = (f_1 f_2 f_9)\cdot(f_3 f_4)\cdot(f_5 f_6 f_7 f_8) & \longrightarrow P_L = \{\{1,2,9\},\{3,4\},\{5,6,7,8\}\} \\
| & \downarrow & \downarrow \\
k & f = (f_1 f_2 f_3 f_4 f_5 f_6 f_7 f_8 f_9) & \longrightarrow P_k = \{\{1,2,3,4,5,6,7,8,9\}\}
\end{array}$$

Notation: Partition P is a *refinement* of Q

$$Q \leq P$$

if each part of Q is a union of parts of P.

Note:

$$L_1 \subseteq L_2 \iff P_{L_1} \leq P_{L_2}$$

Partition P_L example (vectors)

We can encode a partition

$$P_L = \{\{1,2,9\},\ \{3,4\},\ \{5,6,7,8\}\}$$

as $\{0,1\}$-vectors:

$$\begin{aligned} u_1 &= (1,1,0,0,0,0,0,0,1) \\ u_2 &= (0,0,1,1,0,0,0,0,0) \\ u_3 &= (0,0,0,0,1,1,1,1,0) \end{aligned}$$

$$\text{Finding} \quad P_L \quad \iff \quad \text{Finding } U := \text{SPAN}(u_1, u_2, u_3)$$

$$(v_1 \ldots v_9) \in U \iff f_1^{v_1} \cdots f_9^{v_9} \quad \text{is defined over } L$$

$L \rightsquigarrow P_L$ overview

Let $K = k(\alpha)$, let f be the minpoly of α, and $f = f_1 \cdots f_r \in K[x]$. Let L be a subfield of K/k.

How to find: $(v_1, \ldots, v_r) \in \{0,1\}^r$ with $\prod f_i^{v_i} \in L[x]$?

Previous slide: Basis of such vectors $\rightsquigarrow$ Partition P_L

Issue 1: $\prod f_i^{v_i}$ is not linear in the unknowns $v_1, \ldots, v_r$
Solution: use the logarithmic derivative.

Issue 2: Let $h_1, h_2, \ldots$ be coefficients or values of these logarithmic derivatives. We need linear equations for $v_1, \ldots, v_r$ that correspond to $h_1, h_2, \ldots \in L$.
Solution: Use the definition of the i'th principal subfield.

Main issue: efficiency (don't make CPU time worse for small m)
Solution: Two complementary mod p methods.

$L \rightsquigarrow P_L$ overview of efficiency issue

Efficiency issue: The previous slide produces (details later) a large number of equations, with large coefficients in k.

However: The *number* of unknowns, as well as their *values*, are very small (remember we search for $v_1, \ldots, v_r \in \{0, 1\}$!)

Idea: Use only a small subset of the equations, and only compute their images over a finite field.

$\rightsquigarrow$ $\mathcal{O}(r)$ equations over a finite field $\rightsquigarrow$ fast running time.

Question: What about correctness?

Fast probabilistic computation, basic principle

Let M be a 200 by 10 matrix over $k = \mathbb{Q}(t_1, t_2, \sqrt{t_1^3 + 7})$.

Can compute rank(M) with row-reduction. But that is very slow.

Solution:

1. Take a 20 by 10 submatrix (take 20 random rows).
2. Replace t_1, t_2 by random integers.
3. Work mod prime ideal $\rightsquigarrow$ small matrix M_p over a finite field.

We can quickly compute rank(M_p). It probably equals rank(M) but the only thing we know for sure is:

$$\mathrm{rank}(M_p) \leqslant \mathrm{rank}(M).$$

Fast computation of P_L

We have additional methods to determine P_L. Tricks (1),(2),(3) from the previous slide make these methods fast and probabilistic.

But not in the same direction!

With one method we quickly find a partition P and with another[1] method, we quickly find Q in such a way that

$$P \geq P_L \geq Q \quad \text{is provably true}$$

where $\geq$ means refinement of partitions.

If $P = Q$ then we are done.
If not, add more equations (or use another prime ideal).

[1] This explanation omits a third method that is usually faster, but has a more technical proof (Thm. 30 in arXiv:1606.01140)

$L \rightsquigarrow P_L$ details

Let $f = f_1 \cdots f_r \in K[x]$. The i'th principal subfield is:

$$L_i = \{h(\alpha) \mid h(x) \in k[x]_{<n} \text{ and } h(x) \equiv h(\alpha) \bmod f_i\}.$$

We want its partition P_{L_i}. Let

$$g = \prod f_i^{v_i} \text{ and } G = g'/g \text{ (logarithmic derivative)}$$

Take a value of G: $h(\alpha) := G|_{x=c}$ for some $c \in k$. Then

$$h(\alpha) \in L_i \iff h(x) \equiv h(\alpha) \bmod f_i$$

$\rightsquigarrow$ $k-$linear equations for $v_1, \ldots, v_r$

Do this for $2n$ values of G $\rightsquigarrow$ necessary + sufficient equations.

Solve them $\rightsquigarrow$ basis of $\{0,1\}$-vectors $\rightsquigarrow$ partition P_{L_i}

$L \rightsquigarrow P_L$ details, continued

Our algorithm uses $\ll 2n$ values of G.

And: it does not fully compute values + resulting equations, it only computes their images over a finite field.

$\rightsquigarrow$ necessary (but not always sufficient) equations.
$\rightsquigarrow$ a partition $Q \geq P_{L_i}$

Then do another fast (over a finite field) computation, which need not give P_{L_i} either, but it can only fail in the opposite direction ($\leq$ instead of $\geq$)

If both agree, then we provably have P_{L_i} (Las Vegas algorithm)

$\rightsquigarrow$ Quickly find + prove the partition for each principal subfield.

Fast intersections

Recall that $L_1 \subseteq L_2$ if and only if P_{L_2} is a refinement of P_{L_1}.

The partition of $L_1 \bigcap L_2$ is the *join* of partitions P_{L_1} and P_{L_2}, i.e. the finest partition that is refined by both.

Our partitions are only $r \cdot \log(r)$ bits each.
The join of two partitions can be computed quickly [Freese, 1997].

So after computing the partition of each principal subfield, the entire subfield lattice (in terms of partitions) can be computed very quickly, using only $r \cdot \log(r)$ bits of storage per subfield.

Generators of each subfield $P_L \rightsquigarrow L$

Partitions are probably the best way to represent each entry of the subfield lattice. But to compare CPU timings "apples to apples" we also give generators of each subfield.

For each P_L we have to find generators for L.

$$f = f_1 \cdots f_r \in K[x]$$

The partition P_L encodes which $\{f_1, \ldots, f_r\}$-products are in $L[x]$. Take values or coefficients of these products $\rightsquigarrow$ elements of L.

Question: if $h_1, h_2, \ldots \in L$, is there a fast proof they generate L?

Answer: Check if for each principal subfields L_i with $L \not\subseteq L_i$ there is some $h_j \notin L_i$.

That means $h_j(x) \not\equiv h_j(\alpha) \bmod f_i \rightsquigarrow$ give fast proof mod p.

n	r	m	m/r	Magma v2.21-3	Subfields
32	32	374	11.68	11.42s	1.15s
36	16	24	1.50	5.14s	3.84s
50	11	12	1.09	26.06s	24.16s
56	6	6	1.00	52.29s	50.31s
60	18	19	1.05	112.90s	107.53s
60	32	59	1.84	205.46s	118.50s
64	30	93	3.10	167.13s	122.24s
64	64	2,825	44.14	1,084.91s	43.62s
72	24	42	1.75	219.30s	176.65s
75	6	6	1.00	516.45s	542.60
80	27	57	2.11	1,021.22s	685.65s
81	28	56	2.00	715.70s	681.35s
90	7	7	1.00	923.74s	921.77s
96	32	134	4.18	1,159.04s	558.96s
96	56	208	3.71	4,026.65s	2,239.54s
100	57	100	1.75	7,902.09s	4,250.39s
128	128	29,211	228.21	306,591.68s	5,164.75s

Computing decompositions (ISSAC'2017)

Let $f(t) \in k(t)$ be a univariate rational function.

Goal: find **complete decompositions** of f:
indecomposable rational functions $g_1, \ldots, g_m$ such that

$$f = g_1 \circ \cdots \circ g_m.$$

Since: **decompositions** of $f \longleftrightarrow$ **subfields** of $k(t)/k(f(t))$
and: **complete decomp.** of $f \longleftrightarrow$ **max. chains** of subfields

we can use these ingredients:

1. Factor the numerator of $f(t) - f(x)$ as $f_1 \cdots f_r \in k[t, x]$.
2. Principal subfields (vH, Klüners, Novocin) (ISSAC'2011)
3. Fast intersection (Szutkoski, vH) (submitted JSC)
4. Remaining ingredients (ISSAC'2017).

Timings (ISSAC'2017)

k	n	r	$\#dec$	Decompose	Ayad & Fleischmann '08
$\mathbb{F}_{11}$	12	7	3	0.01s	0.03s
$\mathbb{Q}$	24	8	6	0.02s	0.09s
$\mathbb{Q}$	144	10	6	1.82s	101.08s
$\mathbb{F}_{11}$	24	10	8	0.02s	0.20s
$\mathbb{F}_3$	18	12	12	0.05s	0.81s
$\mathbb{F}_{11}$	24	14	12	0.07s	10.57s
$\mathbb{F}_3$	60	17	5	0.18s	981.43s
$\mathbb{Q}$	60	17	5	0.77s	4,338.47s
$\mathbb{F}_{17}$	96	26	44	0.42s	$> 12h$
$\mathbb{F}_{11}$	60	60	111	1.91s	n.a.
$\mathbb{F}_{11}$	120	61	111	2.36s	n.a.
$\mathbb{F}_{13}$	169	91	14	3.41s	n.a.
$\mathbb{F}_5$	120	120	587	18.59s	n.a.
$\mathbb{F}_7$	168	168	680	50.53s	n.a.