Supersingular Isogeny Graphs (in PARI/GP)

James Rickards

Saint Mary's University

james.rickards@smu.ca

26 June 2025

Elliptic curves

- E is an elliptic curve over a field K
- $\mathrm{End}_{\overline{K}}(E) \cong$
  - An order in an imaginary quadratic field
  - or, a maximal order in $B_{p,\infty}$ ($p = \mathrm{char}(K)$ necessarily).
- The second case is *supersingular* (ssl).

Supersingular elliptic curves

- Over $\overline{\mathbb{F}}_p$, all ssl EC's are defined over $\mathbb{F}_{p^2}$
- Isomorphism classes are parametrized by their $j-$invariant, in $\mathbb{F}_{p^2}$
- $\approx \frac{p}{12}$ supersingular isomorphism classes.

$\ell-$isogeny graph

Fix prime p and (prime) ℓ, and make a graph:

- Vertices: isomorphsim classes of ssl EC's, labelled by their $j-$invariant
- Edges: draw an edge if there exists a degree ℓ isogeny between the two curves.
- Often directed: fix representatives, an isogeny is defined up to post-composition by an automorphism (only affects $j = 0, 1728$).

Properties of $\ell-$isogeny graph

- $\ell + 1-$regular if ℓ prime
- Connected
- Ramanujan graph
- Leads to cryptography systems via random walks

How to compute it?

One approach: modular polynomials

- Let $\phi_N(x, y)$ be the N^{th} modular polynomial
- Let $j \in \mathbb{F}_{p^2}$ be supersingular
- For $y \in \mathbb{F}_{p^2}$, $\phi_N(j, y) = 0$ iff y is a ssl $j-$invariant that is $N-$isogenous to j

General algorithm

- Load the N^{th} modular polynomial, reduce modulo p
- Compute one ssl $j-$invariant
- Plug it into ϕ_N, factor over $\mathbb{F}_{p^2}$, repeat!

Modular polynomials

- In PARI/GP for prime level with `polmodular` (computed on the fly)
- Also available from Drew Sutherland's website, $N \leq 400$ and prime $N \leq 1000$
- Save to file, small amount of processing to make it gp-readable with `readvec`

One ssl $j-$invariant

- Find a prime q with $q \equiv 3 \pmod 4$ and $\left(\frac{-q}{p}\right) = -1$ (`forprime`)
- Find Hilbert class polynomial for $-q$ (`polclass`)
- Find a root in $\mathbb{F}_{p^2}$ (`FFX_roots`)

Making the graph

- Depth first search
- Store found $j-$invariants in a `vector`, edges in `vecsmalls`
- Track the $j-$invariants with a hashtable (`hash_init_GEN`, etc.)
- Find the new $j-$invariants with finite field methods (`FFX_factor`, etc.)

Comparison to Sage

- Same general algorithm is implemented in SageMath
- Restricts to ℓ prime and is very slow, but also handles the non-supersingular case
- I also provide code to easily call the method from Sage

ℓ	Approximate speedup from `E.isogeny_ell_graph`
2	67
3	107
5	211
7	217

Table 1: Timings from PARI/GP 2.16.1 and SageMath 10.2. Average factor of improvement for 20 primes between 100 and 3000.

Comments

- Sage integration: `gen_to_sage` is VERY slow
- `pari(code)` is significantly faster
- Sage integration relies on `ffgen(p^2)` always outputting the same generator

`https://github.com/JamesRickards-Canada/Isogeny`