Supersingular isogeny graphs and traces of endomorphisms

Travis Morrison

December 9, 2018

Here is some sample usage for the functions isogeny_graph(p, ell) and trace_of_chain(chain). We construct the isogeny graph:

```
sage: load(``isogney_graphs.sage'')
sage: Gpell, Gpell_graph = isogeny_graph(157,3)
```

The output Gpell_graph is a Graph object in sage, so you can use Gpell_graph.adjacency_matrix() or Gpell_graph.show() to visualize it, for example. The output Gpell stores the curves and isogenies used in constructing the graph.

```
sage: F = GF(157^2)
sage: z2 = F.gen()
sage: Gpell[F(79)]['curve']
Elliptic Curve defined by y^2 = x^3 + 40*x + 39 over Finite Field in z2
of size 157^2
sage: Gpell[F(79)]['neighbors'].keys()
[68*z2 + 62, 80*z2 + 100, 89*z2 + 88, 77*z2 + 29]
sage: Gpell[F(79)]['neighbors'][68*z2+62]
[Isogeny of degree 3 from Elliptic Curve defined by y^2 = x^3 + 40*x +
39 over Finite Field in z2 of size 157^2 to Elliptic Curve defined by
y^2 = x^3 + (82*z2+85)*x + (74*z2+137) over Finite Field in z2 of size
157^2]
```

We compute the trace of an endomorphism of degree 3^3 represented by a chain of 3-isogenies in $G(157, 3)$:

```
sage: F = GF(157^2)
sage: z2 = F.gen()
sage: path = [F(79), 89*z2+88, 68*z2+62, F(79)]
sage: chain = path_to_chain(path,Gpell)
sage: trace = trace_of_chain(chain)
sage: print(trace)
2
```

We check that the trace is correct:

```
sage: E = chain[0].domain()
sage: P = E.random_point()
sage: phiP = evaluate_chain(chain,P)
sage: phiphiP = evaluate_chain(chain,phiP)
sage: phiphiP - trace*phiP + 27*P == E(0)
True
```