Counting points on curves in average polynomial time

David Harvey and Andrew Sutherland

February 24, 2014

http://arxiv.org/abs/1402.3246

Sato-Tate in genus 1

Let $E/\mathbb{Q}$ be an elliptic curve:

$$y^2 = x^3 + Ax + B.$$

Let p be a prime of good reduction for E.
The number of $\mathbf{F}_p$-points on the reduction E_p of E modulo p is

$$\#E_p(\mathbf{F}_p) = p + 1 - t_p.$$

The trace of Frobenius t_p is an integer in the interval $[-2\sqrt{p}, 2\sqrt{p}]$.

We are interested in the limiting distribution of the *normalized* value

$$x_p = \frac{-t_p}{\sqrt{p}} \in [-2, 2],$$

as p varies over primes of good reduction.

Zeta functions and L-polynomials

For a smooth projective curve $C/\mathbb{Q}$ of genus g and a good prime p let

$$Z(C_p/\mathbb{F}_p;T) := \exp\left(\sum_{k=1}^{\infty} N_k T^k/k\right),$$

where $N_k = \#C_p(\mathbf{F}_{p^k})$. This is a rational function of the form

$$Z(C_p/\mathbb{F}_p;T) = \frac{L_p(T)}{(1-T)(1-pT)},$$

where $L_p(T)$ is an integer polynomial of degree $2g$.

For $g=1$ we have $L_p(t) = pT^2 + c_1T + 1$, and for $g=2$,

$$L_p(T) = p^2T^4 + c_1pT^3 + c_2T^2 + c_1T + 1.$$

Sato-Tate in genus g

The normalized L-polynomial

$$\bar{L}_p(T) := L_p(T/\sqrt{p}) = \sum_{i=0}^{2g} a_i T^i \in \mathbf{R}[T]$$

is monic, symmetric ($a_i = a_{2g-i}$), and unitary (roots on the unit circle). The coefficients a_i necessarily satisfy $|a_i| \leq \binom{2g}{i}$.

We may now consider the limiting distribution of $a_1, a_2, \ldots, a_g$ over all primes $p \leq N$ of good reduction, as $N \to \infty$.

`http://math.mit.edu/~drew`

Existing algorithms for hyperelliptic curves

Algorithms to compute $L_p(T)$ for low genus hyperelliptic curves:

	complexity (ignoring factors of $O(\log\log p)$)		
algorithm	$g=1$	$g=2$	$g=3$

Existing algorithms for hyperelliptic curves

Algorithms to compute $L_p(T)$ for low genus hyperelliptic curves:

	complexity (ignoring factors of $O(\log\log p)$)		
algorithm	$g=1$	$g=2$	$g=3$
point enumeration	$p\log p$	$p^2\log p$	$p^3\log p$

Existing algorithms for hyperelliptic curves

Algorithms to compute $L_p(T)$ for low genus hyperelliptic curves:

	complexity (ignoring factors of $O(\log\log p)$)		
algorithm	$g = 1$	$g = 2$	$g = 3$
point enumeration	$p\log p$	$p^2\log p$	$p^3\log p$
group computation	$p^{1/4}\log p$	$p^{3/4}\log p$	$p^{5/4}\log p$

Existing algorithms for hyperelliptic curves

Algorithms to compute $L_p(T)$ for low genus hyperelliptic curves:

algorithm	complexity (ignoring factors of $O(\log\log p)$) $g = 1$	$g = 2$	$g = 3$
point enumeration	$p \log p$	$p^2 \log p$	$p^3 \log p$
group computation	$p^{1/4} \log p$	$p^{3/4} \log p$	$p^{5/4} \log p$
p-adic cohomology	$p^{1/2} \log^2 p$	$p^{1/2} \log^2 p$	$p^{1/2} \log^2 p$

Existing algorithms for hyperelliptic curves

Algorithms to compute $L_p(T)$ for low genus hyperelliptic curves:

	complexity (ignoring factors of $O(\log\log p)$)		
algorithm	$g=1$	$g=2$	$g=3$
point enumeration	$p\log p$	$p^2\log p$	$p^3\log p$
group computation	$p^{1/4}\log p$	$p^{3/4}\log p$	$p^{5/4}\log p$
p-adic cohomology	$p^{1/2}\log^2 p$	$p^{1/2}\log^2 p$	$p^{1/2}\log^2 p$
CRT (Schoof-Pila)	$\log^5 p$	$\log^8 p$	$\log^{12} p$ (?)

Existing algorithms for hyperelliptic curves

Algorithms to compute $L_p(T)$ for low genus hyperelliptic curves:

	complexity (ignoring factors of $O(\log\log p)$)		
algorithm	$g = 1$	$g = 2$	$g = 3$
point enumeration	$p\log p$	$p^2\log p$	$p^3\log p$
group computation	$p^{1/4}\log p$	$p^{3/4}\log p$	$p^{5/4}\log p$
p-adic cohomology	$p^{1/2}\log^2 p$	$p^{1/2}\log^2 p$	$p^{1/2}\log^2 p$
CRT (Schoof-Pila)	$\log^5 p$	$\log^8 p$	$\log^{12} p$ (?)

An average polynomial-time algorithm

All of the methods above perform separate computations for each p. But we want to compute $L_p(T)$ for all good $p \leq N$ using reductions of *the same curve* in each case.

An average polynomial-time algorithm

All of the methods above perform separate computations for each p. But we want to compute $L_p(T)$ for all good $p \le N$ using reductions of *the same curve* in each case.

Theorem (H 2012)

There exists a deterministic algorithm that, given a hyperelliptic curve $y^2 = f(x)$ of genus g with a rational Weierstrass point and an integer N, computes $L_p(T)$ for all good primes $p \le N$ in time

$$O\left(g^{8+\epsilon} N \log^{3+\epsilon} N\right),$$

assuming the coefficients of $f \in \mathbf{Z}[x]$ have size bounded by $O(\log N)$.

Average time is $O\left(g^{8+\epsilon} \log^{4+\epsilon} N\right)$ per prime, polynomial in g and $\log p$. Very recently (last week) generalized to arithmetic schemes.

An average polynomial-time algorithm

But is it practical?

An average polynomial-time algorithm

But is it practical? Yes!

algorithm	complexity (ignoring factors of $O(\log\log p)$) $g=1$	$g=2$	$g=3$
point enumeration	$p\log p$	$p^2\log p$	$p^3\log p$
group computation	$p^{1/4}\log p$	$p^{3/4}\log p$	$p^{5/4}\log p$
p-adic cohomology	$p^{1/2}\log^2 p$	$p^{1/2}\log^2 p$	$p^{1/2}\log^2 p$
CRT (Schoof-Pila)	$\log^5 p$	$\log^8 p$	$\log^{12} p(?)$
Average polytime	$\log^4 p$	$\log^4 p$	$\log^4 p$

	$d=5$		$d=6$	
N	ave polytime	group comp	ave polytime	group comp
2^{14}	0.4	0.2	0.7	0.3
2^{15}	1.1	0.6	1.9	0.7
2^{16}	2.8	1.7	4.9	2.0
2^{17}	6.8	5.6	11.9	6.4
2^{18}	16.8	20.2	29.0	22.1
2^{19}	39.7	76.4	69.1	83.4
2^{20}	94.4	257	166	284
2^{21}	227	828	398	914
2^{22}	534	2630	946	2900
2^{23}	1240	8570	2230	9520
2^{24}	2920	28000	5260	31100
2^{25}	6740	92300	11800	102000
2^{26}	15800	316000	27400	349000

Performance comparison of new algorithm (ave polytime) with `smalljac` (group comp) in genus 2. Times in CPU seconds.

	$d = 7$	
N	ave polytime	p-adic
2^{14}	2.0	6.8
2^{15}	5.5	15.6
2^{16}	13.6	37.6
2^{17}	33.3	95.0
2^{18}	80.4	250
2^{19}	192	681
2^{20}	459	1920
2^{21}	1090	5460
2^{22}	2540	16300
2^{23}	5940	49400
2^{24}	13800	152000
2^{25}	31800	467000
2^{26}	72900	1490000

Performance comparison of new algorithm (ave polytime) with `hypellfrob` (p-adic) in genus 2. Times in CPU seconds.

The algorithm in genus 1

The Hasse invariant h_p of an elliptic curve $y^2 = f(x) = x^3 + ax + b$ over $\mathbf{F}_p$ is the coefficient of x^{p-1} in the polynomial $f(x)^{(p-1)/2}$.

We have $h_p \equiv t_p \bmod p$, which uniquely determines t_p for $p > 13$.

Naïve approach: iteratively compute $f, f^2, f^3, \ldots, f^{(N-1)/2}$ in $\mathbf{Z}[x]$ and reduce the x^{p-1} coefficient of $f(x)^{(p-1)/2}$ mod p for each prime $p \leq N$.

The algorithm in genus 1

The Hasse invariant h_p of an elliptic curve $y^2 = f(x) = x^3 + ax + b$ over $\mathbf{F}_p$ is the coefficient of x^{p-1} in the polynomial $f(x)^{(p-1)/2}$.

We have $h_p \equiv t_p \bmod p$, which uniquely determines t_p for $p > 13$.

Naïve approach: iteratively compute $f, f^2, f^3, \ldots, f^{(N-1)/2}$ in $\mathbf{Z}[x]$ and reduce the x^{p-1} coefficient of $f(x)^{(p-1)/2}$ mod p for each prime $p \leq N$.

But the polynomials f^n are huge, each has $\Omega(n^2)$ bits.
It would take $\Omega(N^3)$ time to compute $f, \ldots, f^{(N-1)/2}$ in $\mathbf{Z}[x]$.

So this is a terrible idea...

The algorithm in genus 1

The Hasse invariant h_p of an elliptic curve $y^2 = f(x) = x^3 + ax + b$ over $\mathbf{F}_p$ is the coefficient of x^{p-1} in the polynomial $f(x)^{(p-1)/2}$.

We have $h_p \equiv t_p \bmod p$, which uniquely determines t_p for $p > 13$.

Naïve approach: iteratively compute $f, f^2, f^3, \ldots, f^{(N-1)/2}$ in $\mathbf{Z}[x]$ and reduce the x^{p-1} coefficient of $f(x)^{(p-1)/2}$ mod p for each prime $p \leq N$.

But the polynomials f^n are huge, each has $\Omega(n^2)$ bits.
It would take $\Omega(N^3)$ time to compute $f, \ldots, f^{(N-1)/2}$ in $\mathbf{Z}[x]$.

So this is a terrible idea...

But we don't need all the coefficients of f^n, we only need one;
and we only need to know its value modulo $p = 2n + 1$.

A better approach

Let $f(x) = x^3 + ax + b$, and let f_k^n denote the coefficient of x^k in $f(x)^n$. Using $f^n = f \cdot f^{n-1}$ and $(f^n)' = nf'f^{n-1}$, one obtains the relations

$$\begin{aligned}(n+2)f_{2n-2}^n &= n\left(2af_{2n-3}^{n-1} + 3bf_{2n-2}^{n-1}\right),\\ (2n-1)f_{2n-1}^n &= n\left(3f_{2n-4}^{n-1} + af_{2n-2}^{n-1}\right),\\ 2(2n-1)bf_{2n}^n &= (n+1)af_{2n-4}^{n-1} + 3(2n-1)bf_{2n-3}^{n-1} - (n-1)a^2f_{2n-2}^{n-1}.\end{aligned}$$

These allow us to compute the vector $v_n = [f_{2n-2}^n, f_{2n-1}^n, f_{2n}^n]$ from the vector $v_{n-1} = [f_{2n-4}^{n-1}, f_{2n-3}^{n-1}, f_{2n-2}^{n-1}]$ via multiplication by a 3×3 matrix M_n with entries in $\mathbf{Q}$. We have

$$v_n = v_0 M_1 M_2 \cdots M_n.$$

For $n = (p-1)/2$, the Hasse invariant of the elliptic curve $y^2 = f(x)$ over $\mathbf{F}_p$ is obtained by reducing the third entry f_n^{2n} of v_n modulo p.

Computing $t_p \bmod p$

To compute $t_p \bmod p$ for all odd primes $p \le N$ it suffices to compute

$$\begin{aligned} M_1 &\bmod 3 \\ M_1 M_2 &\bmod 5 \\ M_1 M_2 M_3 &\bmod 7 \\ M_1 M_2 M_3 M_4 &\bmod 9 \\ &\vdots \\ M_1 M_2 M_3 \cdots M_{(N-1)/2} &\bmod N \end{aligned}$$

Doing this naïvely would take $O\big(N^{2+\epsilon}\big)$ time.
But it can be done in $O\big(N^{1+\epsilon}\big)$ time using a *remainder tree*.

The algorithm in genus g.

The *Hasse-Witt* matrix of a hyperelliptic curve $y^2 = f(x)$ over $\mathbf{F}_p$ of genus g is the $g \times g$ matrix $W_p = [w_{ij}]$ with entries

$$w_{ij} = f_{pi-j}^{(p-1)/2} \bmod p.$$

The algorithm in genus g.

The *Hasse-Witt* matrix of a hyperelliptic curve $y^2 = f(x)$ over $\mathbf{F}_p$ of genus g is the $g \times g$ matrix $W_p = [w_{ij}]$ with entries

$$w_{ij} = f_{pi-j}^{(p-1)/2} \bmod p.$$

The w_{ij} can each be computed using recurrence relations between the coefficients of f^n and those of f^{n-1}, as in genus 1.

The congruence

$$L_P(T) \equiv \det(I - TW_p) \bmod p$$

allows us to determine the coefficients $a_1, \ldots, a_g$ of $L_p(T)$ modulo p.

Using group computations in the Jacobian of the curve, one can determine $L_p(T)$ exactly. This takes $\tilde{O}(1)$ time in genus 2, and $\tilde{O}(p^{1/4})$ time in genus 3, which turns out to be negligible within the feasible range of computation.

Optimizations

The remainder tree algorithm can be made faster and more space efficient using a *remainder forest*.

Our implementation works for all hyperelliptic curves, not just those with a rational Weierstrass point.

Theorem (HS 2014)

There exists a deterministic algorithm that, given a hyperelliptic curve $y^2 = f(x)$ of genus g and an integer N, computes $L_p(T)$ for all good primes $p \le N$ using

$$O\left(g^5 N \log^{3+\epsilon} N\right) \text{ time} \qquad \text{and} \qquad O(g^2 N) \text{ space},$$

assuming that g and the size of the coefficients of $f \in \mathbf{Z}[x]$ are $O(\log N)$.