Optimized equations for $X_1(N)$ via simulated annealing

Peter Caday and Andrew V. Sutherland
Department of Mathematics, Massachusetts Institute of Technology

1. Background

Some models of an algebraic curve C are better than others. In particular, low-degree models are useful:

- *practically,* to locate points on C quickly;
- *theoretically,* to prove upper bounds on the gonality of C.

Given a defining equation $F(r,s)=0$ for C, we seek a bilinear transformation that reduces the degree of F in one of its variables. Our motivating example is the modular curve $X_1(N)$, which parameterizes elliptic curves with a point of order N. A general method to obtain a defining equation $F(r,s)=0$ for $X_1(N)$ (more precisely, the affine curve $Y_1(N)$) is given in [4]. Roots of F may then be used to construct elliptic curves with a point of order N (over a finite field, say), an idea exploited in [6]. The efficiency of this construction depends critically on F, which is typically larger and of higher degree than necessary. We wish to construct an optimized equation $f(x,y)=0$, together with an explicit birational map ϕ that relates the roots of f and F.

Some suitable choices for f are given in [3, 4] for $N \le 18$, in [7] for $N \le 22$, and in [1] for $N \le 51$. We have developed a new algorithm that uses simulated annealing to find optimized models for arbitrary algebraic curves. This algorithm has been successfully applied to $X_1(N)$ for $N \le 101$, matching and in many cases improving these results.

2. Transformation of the Problem

Working by hand, one can often simplify $F(r,s)=0$ by applying suitable series of translations and inversions, in an attempt to remove singularities. To automate this process, we fix a small set of invertible atomic operations $\phi_i : \mathbb{A}^2 \to \mathbb{A}^2$ of this kind: the maps $(x,y) \mapsto (x+a,y)$, $(x,y) \mapsto (1/x,y)$ and $(x,y) \mapsto (1/x, y/x)$ are three typical examples.

Let $\mathcal{C}$ be the set of curves obtained by applying a finite number of these atomic operations to the curve C_0 defined by the given polynomial $F_0 = F$. We think of $\mathcal{C}$ as the vertices of a graph G, with edges $(C, \phi_i \circ C)$. A path in G defines a birational map ϕ; reversing the path yields the inverse map.

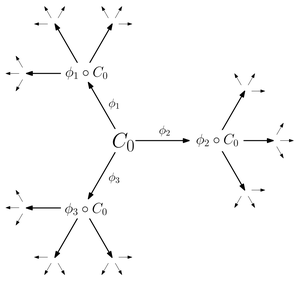


Figure 1: The search graph G

As stated above, our problem is to find a curve in $\mathcal{C}$ whose defining polynomial has minimal degree in one of its variables, say y. Additionally, we might favor polynomials with fewer terms or smaller coefficients. To hide these details, and for the sake of generality, we choose a *cost function* k on $\mathcal{C}$ so that $k(C_1) < k(C_2)$ whenever C_1 is more desirable than C_2. Our original problem of finding a better model for a given algebraic curve is now a combinatorial optimization problem: we seek a low-cost vertex in G.

3. Simulated Annealing

The algorithm in [5] finds low-cost vertices in G by exhaustively searching a local neighborhood. Our new algorithm uses simulated annealing, a widely used optimization method introduced by Kirkpatrick et al. in [2]. Simulated annealing is a local search algorithm: starting at the origin C_0, it selects a neighboring vertex C_1, moves to it, and repeats.

The key, of course, is picking C_1. The simplest approach is to select a neighbor at random, and move to it only if it has a lower cost. Simulated annealing extends this approach, also allowing moves to higher-cost neighbors with an appropriate probability, to avoid becoming trapped at a local minimum. This probability is dependent on the cost difference as well as a *temperature* parameter T that decays during the course of the algorithm. More precisely, the probability of moving from C_1 to C_2 is

$$p(C_1, C_2) = \max\left(e^{\frac{k(C_1)-k(C_2)}{T}}, 1\right).$$

When T is large, $p(C_1,C_2)$ can be high even when $k(C_2)$ is somewhat larger than $k(C_1)$. As T approaches zero, the probability of moving to a vertex of higher cost likewise approaches zero.

Our initial implementation of the simulated annealing approach was good at finding low-cost vertices in G, but the path used to reach them was often much longer than necessary. This made the corresponding birational map ϕ more complicated than we would like, potentially requiring thousands of atomic operations. To address this problem, we introduced the possibility of *backtracking* along the path from C_0 to the current vertex. With aggressive backtracking, and by limiting the search to vertices within a bounded distance of C_0, we achieve comparable results with fewer than 80 atomic operations.

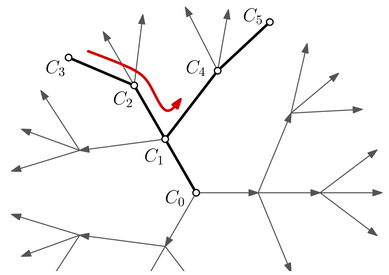


Figure 2: Backtracking during a simulated annealing search; $C_0, C_1, \ldots$ are visited successively

4. Example

Table 1 illustrates the algorithm applied to the "raw" model of $X_1(13)$ from [4]. The final, simplified polynomial was obtained by applying nine atomic operations to this raw model. The rows of the table correspond to successive vertices on the path from the initial to the final polynomial, omitting backtracks, starting with the initial polynomial $F_0(x,y)$. For brevity, some intervening polynomials have been omitted.

The values T_i are the values of the temperature parameter at the transition from C_{i-1} to C_i, and $p(C_{i-1}, C_i)$ is the probability of acceptance for this transition defined earlier.

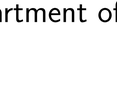
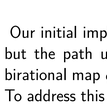
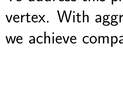
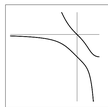
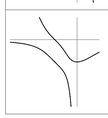
C_i	$k(C_i)$	T_i	$p(C_{i-1}, C_i)$	$F_i(x,y)$
	28.28	–	–	$x^3-x^2y^4+5x^2y^3-9x^2y^2+4x^2y-2x^2-xy^3+6xy^2-3xy+x-y^3$
	28.95	1.000	0.51	$x^3y^4 - x^2 + 5x^2y - 9x^2y^2 + 4x^2y^3 - 2x^2y^4 - xy + 6xy^2 - 3xy^3 + xy^4 - y$
	28.58	0.998	1	$y^4 - x + 5xy - 9xy^2 + 4xy^3 - 2xy^4 - x^2y + 6x^2y^2 - 3x^2y^3 + x^2y^4 - x^3y$
	28.61	0.998	0.97	$-x^3y + x^2y^4 - 3x^2y^3 + 6x^2y^2 - 4x^2y - 2xy^3 + 3xy^2 - x + y^3 - 3y^2 + 3y - 1$
⋮				
	19.27	0.624	1	$x^3y - 2x^2y + x + xy - xy^2 + y + y^2$
	19.23	0.583	1	$x^3y + x^2y - xy^2 + x + y + 1$

Table 1: Sample run for $X_1(13)$

5. Results

5.1. Models of $X_1(N)$.

We applied the new algorithm to the modular curves $X_1(N)$, for $N \le 101$, aiming to obtain plane models that minimize degree (in each variable), total degree, number of terms, and coefficient size. Table 1 provides examples of the results for selected values of N. The integers $d_y(C_0)$ and $d_y(C_1)$ are the degrees in y of the initial and final curves, respectively, while $t(C_0)$ and $t(C_1)$ represent the number of terms for each.

N	$d_y(C_0)$	$d_y(C_1)$	$t(C_0)$	$t(C_1)$
26	8	6	82	27
40	19	15	412	171
49	39	31	1791	661
62	48	37	2666	925
75	80	63	7567	2648
88	96	76	10730	3836
101	170	134	33618	11801

Table 2: Original and final models of $X_1(N)$

We note that the birational map ϕ output by the algorithm was the same for many values of N, and often remarkably simple. For example the map

$$r = \frac{x^2(x+1)+(x+y)}{-xy(x+1)+(x+y)}, \qquad s = \frac{x^2+(x+y)}{-xy+(x+y)},$$

was the best obtained for $55 \le N \le 101$. This suggests that for larger values of N, a quick and easy optimization is to simply apply this map to the raw form of $X_1(N)$ defined in [5].

5.2. Runtime.

The major advantage of the present algorithm over our previous efforts is its speed. The algorithm given in [5] requires approximately R^8 atomic operations where typically $R \ge 8$. On the other hand, a single iteration of simulated annealing typically requires only 10,000 atomic operations, although multiple iterations are often useful. As a result, simulated annealing is practical even for very large polynomials such as the raw form of $X_1(101)$, which has total degree 352 and $33{,}618$ terms.

N	original algorithm	simulated annealing
21	11.4	0.08
31	116.3	0.43
41	1116	1.65
51	1485	2.30
61	–	10.6
71	–	30.2
81	–	36.1
91	–	111.1
101	–	162.2

Table 3: Timings for $X_1(N)$ in seconds on a 2.66GHz Xeon core

The asymptotic running time of the simulated annealing algorithm is $O(\ell d^3)$, where ℓ is the distance between the initial and final polynomials in atomic operations, and d is the maximum total degree over all polynomials examined. It appears difficult to overcome the $O(d^3)$ barrier within our present framework, since this is the time required to translate polynomials.

On the other hand, we have been able to improve performance using modular arithmetic, working modulo some large prime p rather than over the integers. In this scenario we obtain our final result by applying the birational map ϕ to the initial curve. We also find that it is better to use multiple fast rounds of simulated annealing in place of a single slow round.

References

[1] Houria Baaziz, *Equations for the modular curve $X_1(N)$ and models of elliptic curves with torsion points*, Mathematics of Computation, **79** (2010), no. 272, 2371–2386.

[2] S. Kirkpatrick, C. D. Gelatt and M. P. Vecchi. *Optimization by simulated annealing*. Science, New Series **220** (1983), no. 4598, pp. 671–680.

[3] Daniel Sion Kubert, *Universal bounds on the torsion of elliptic curves*, Proceedings of the London Mathematical Society **33** (1976), 193–237.

[4] Markus A. Reichert, *Explicit determination of nontrivial torsion structures of elliptic curves over quadratic number fields*, Mathematics of Computation **46** (1986), no. 174, 637–658.

[5] Andrew V. Sutherland, *Constructing elliptic curves over finite fields with prescribed torsion*, 2008, `http://arxiv.org/abs/0811.0296`.

[6] Andrew V. Sutherland, *Computing Hilbert class polynomials with the Chinese Remainder Theorem*, Mathematics of Computation, posted on May 17, 2010, PII S 0025-5718(2010)02373-7 (to appear in print).

[7] Yifan Yang, *Defining equations for modular curves*, Advances in Mathematics **204** (2006), no. 2, 481–508.