ACM Communications in Computer Algebra, Vol 47, No. 3, Issue 185, September 2013

Recursive Sparse Interpolation

Andrew Arnold Mark Giesbrecht
Symbolic Computation Group
University of Waterloo, Canada
{a4arnold,mwg}@uwaterloo.ca

Dan Roche
United States Naval Academy, USA
roche@usna.edu

We consider the problem of interpolating a sparse univariate polynomial f over an arbitrary ring, given by a straight-line program. In this problem we are given a straight-line program that computes f, as well as bounds D and T on the degree and sparsity (i.e., the number of nonzero terms) of f respectively. We build on ideas developed in Garg and Schost (2009) and Giesbrecht and Roche (2011) towards algorithms for this specific problem. We present a Monte Carlo algorithm that improves on the best previously-known algorithm for this specific problem by a factor (softly) on the order of $T/\log D$. Thus this new algorithm is favourable for "moderate" values of T.

Our algorithm is recursive. At a recursive step of the algorithm we have a straight-line program for f, an approximation f^* of f, and respective bounds T and D on the sparsity and degree of the difference $g = f - f^*$. We initialize f^* to zero. We will construct an approximation f^{**} to g such that, with high probability, $g - f^{**}$ has at most $T/2$ terms. We then recurse with $f^* + f^{**}$ as our refined approximation for f.

The algorithms in Garg and Schost (2009) and Giesbrecht and Roche (2011), as well as the algorithm we will present, interpolate f by using its straight-line program to evaluate f at a symbolic k-th root of unity, for appropriate choices of k. This effectively gives the image $f \bmod (z^k - 1)$. We call such an evaluation a *probe* of degree k. The cost of a degree-k probe to a length-L straight-line program is quasi-linear in kL. We use the number of probes, multiplied by a bound on the probe degree, as a rough measure of the cost of an interpolation algorithm.

The image $f \bmod (z^k - 1)$ in practise gives a large amount of useful information about the polynomial f. Namely, a term cz^e of f will appear as $cz^{e \bmod k}$ in the image $f \bmod (z^k - 1)$, so the image should give us f's vector of exponents modulo k. However, there are potential obstacles. We may not be able to match images of the same term in multiple images of f. In addition, terms can *collide* modulo $z^k - 1$ if they have the same degree modulo k. Collisions are problematic because it is difficult to detect if a term in an image $f \bmod (z^k - 1)$ is in fact the image of a sum of colliding terms. Alternatively, colliding terms may sum to zero modulo $z^k - 1$, which also may be difficult to detect.

Previous Las Vegas interpolation algorithms require a "good" prime, a prime p for which the terms of f remain distinct modulo $z^p - 1$. If p is a good prime, $f \bmod (z^p - 1)$ has the same number of terms as f. Thus, once we have a good prime with high probability, we can detect the presence of collisions in other images of f. In order to guarantee one can find such a prime with high probability, one chooses primes at random on the order of $T^2 \log D$ as probe degrees.

In order to reduce this probe degree, we relax the condition that p separates all the terms of the difference g. We instead look for an "ok" prime: a prime which separates *most* of the terms of g. This allows instead to search over primes p of size $\mathcal{O}(T \log D)$.

Once we have an "ok" prime, we make probes of degree pq_i for a set of co-prime q_i, each of size $\mathcal{O}(\log D)$. Our probe degree thus becomes $\mathcal{O}(T \log^2 D)$. If a term of g does not collide with another term modulo $z^p - 1$ then it will not collide modulo $(z^{pq_i} - 1)$. These probes will allow us to construct a polynomial f^{**} containing the non-colliding terms of g, plus potentially a small proportion of deceptive terms: terms

constructed from garbage information due to collisions in the images $f \bmod (z^{pq_i} - 1)$. Fortunately, if p is an ok prime we can give an upper bound on the number of such deceptive terms that can appear in f^{**}.

After we construct f^{**} we then recursively interpolate the new difference $g - f^{**}$, with a new sparsity bound $T/2$. We continue in this fashion $\lfloor \log T \rfloor + 1$ times until the sparsity bound reaches 0. An advantage of the recursive nature of the algorithm is that, when we reach a threshold where $\log D$ begins to dominate T, we can plug in a better-suited algorithm to interpolate what remains.

References

Sanchit Garg and Éric Schost. Interpolation of polynomials given by straight-line programs. *Theor. Comput. Sci.*, 410(27-29):2659–2662, June 2009. ISSN 0304-3975. doi: 10.1016/j.tcs.2009.03.030. URL `http://dx.doi.org/10.1016/j.tcs.2009.03.030`.

M. Giesbrecht and D.S. Roche. Diversification improves interpolation. *ISSAC '11*, pages 123–130, 2011. doi: 10.1145/1993886.1993909. URL `http://doi.acm.org/10.1145/1993886.1993909`.