Polynomial Factorization Challenges

Paul Zimmermann
Inria Lorraine
zimmermann@loria.fr

Laurent Bernardin
ETH Zürich
bernardin@inf.ethz.ch

Michael Monagan
Simon Fraser University
monagan@cecm.sfu.ca

July 1996

Abstract

Joachim von zur Gathen has proposed a challenge for factoring univariate polynomials over finite fields to evaluate the practicability of current factorization algorithms ("A Factorization Challenge", SIGSAM Bulletin 26(2):22-24, 1992). More recently, Victor Shoup has proposed an alternate family of polynomials with a similar goal in mind.

Our effort is to take these challenges on using the general purpose computer algebra systems Maple and MuPAD.

The result of our work are the factorizations of the von zur Gathen polynomials f_n and of the Shoup polynomials F_n for n from 1 to 500. We also present the factorization of the degree 1000 von zur Gathen polynomial f_{1000}.

1 The Algorithms

We implemented Shoup's distinct degree factorization algorithm [4] in Maple and MuPAD. For the equal degree splitting, Maple uses the standard Cantor-Zassenhaus algorithm while MuPAD implements an improved version that uses some of the intermediate computations from the distinct degree step.

Because we are working with large coefficients, it is crucial that the modular integer arithmetic is as fast as possible. MuPAD uses PARI [1] and proves to be a factor of 2 faster than Maple's proprietary integer arithmetic. In an experimental version of Maple we used Karatsuba multiplication for large integers to alleviate this problem somewhat.

For univariate polynomials MuPAD uses a sparse data structure. By contrast, Maple uses a dense representation [2] which uses lazy reduction techniques to keep the number of integer divisions linear in the degree of the polynomials for multiplication and division. Additionally, we have implemented Karatsuba multiplication for univariate polynomials in an experimental version.

2 The von zur Gathen Polynomials

Von zur Gathen's challenge involves factoring the polynomials

$$f_n = x^n + x + 1 \pmod{p_n}$$

where p_n is the smallest prime larger than $2^n\pi$. The coefficients of the polynomial f_n has thus a size of $n-2$ bits. In [3] Mike Monagan presented the factorization of the von zur Gathen polynomials up to $n = 200$.

3 The Shoup Polynomials

In [4] Victor Shoup proposes to use the following family of polynomials for comparing univariate factorization algorithms

$$F_n = \sum_{i=0}^{n} a_i x^{n-i} \pmod{P_n}$$

where $a_0 = 1$ and $a_{i+1} = a_i^2 + 1$. P_n is the smallest prime larger than $2^{n-2}\pi$. This family of polynomials has the advantage that its members are dense and the coefficients of F_n are n bits large.

4 Results

We have factored f_n and F_n for n up to 500, both with Maple and MuPAD. The degree sequences can be found on
`http://www.loria.fr/~zimmerma/mupad/` and the complete factorizations on
`http://www.inf.ethz.ch/personal/bernardi/challenge/`.

We have also factored f_{1000}, the degree 1000 von zur Gathen polynomial. For this we used our experimental version of Maple which included Karatsuba multiplication algorithms for both integers and polynomials. The factorization took 5 days of CPU time on a DEC Alpha (with a 200 Mhz clock). The degree pattern that we computed is $[1, 7, 32, 960]$. We were however not yet able to verify the irreducibility of the degree 960 factor using a different algorithm or implementation.

5 Discussion

We have done extensive timing tests to compare our implementations of Shoup's algorithm to the Cantor-Zassenhaus and to the Big Prime Berlekamp algorithms, in order to find the break-even point of these algorithms. In the MuPAD implementation, both for f_n and F_n, Shoup's algorithm seem to be more efficient than the default algorithm (Cantor-Zassenhaus) for $n \geq 15$. For Maple the break even with the Cantor-Zassenhaus algorithm is around $n = 20$. We also did a comparison with respect to both the degree n of the polynomial and the base-2 logarithm $\lg p$. It appears that Shoup's algorithm is faster than Cantor-Zassenhaus for $n \geq 6$ and $\lg p \geq 15$ in the MuPAD implementation.

As mentionned before, the Maple implementation of univariate polynomials over prime fields uses a special data structure. For large moduli ($p > \sqrt{2^{16}}$), the representation is an array of multiprecision integers. Multiplication and Divison use lazy reduction techniques such that the number of integer divisons remains linear in the degree of the polynomials. As a consequence, a fast algorithm for multiplying multiprecision integers is more important than a fast division algorithm.

For polynomial arithmetic we have implemented Karatsuba multiplication. As the underlying coefficient arithmetic becomes more and more expensive with an increasing modulus, the overhead of Karatsuba becomes less and less important. With a large enough modulus we can reach a break even of 2. But even if a Karatsuba multiplication gives a good efficiency improvement of the factorization, we also need a fast division algorithm, based on inverse power series or Newton iteration. We have not yet implemented such an algorithm but present timings illustrating the impact of Karatsuba multiplication.

Note that computations, which like the ones presented here require arithmetic with multiprecision integers, are very sensitive on the wordsize of the CPU being used. Going from a 32 bit to a 64 bit architecture, we notice that much fewer machine operations are required to achieve the same multiprecision result. Classical multiplication is speeded up by a factor of 4. Karatsuba multiplication improves by a factor of 3. We verified this by comparing timings on a Sparcstation 20/51 (32 bit) and a DEC Alpha (64 bit).

6 Conclusions

We have shown how far two general purpose computer algebra systems can tackle the polynomial factorization challenges. Assuming that further improvements on the algorithmic level we have seen two directions for stretching the limits of these systems even more; One possibility is to improve the basic aritmetic on integers and polynomials by implementing FFT based multiplication and division algorithms. However, such improvements will only profit a reduced set of applications and may not be reasonable for a general purpose computer algebra system. The second direction is improvements of the computer hardware. We have seen that doubling the wordsize of a processor has increased the performance of the factorization algorithms by a factor of three to four. With the emergence of 128 bit architectures we can expect a similar increase.

References

[1] Batut, C., Bernardi, D., Cohen, H., and Olivier, M. *User's Guide to PARI-GP*, Jan. 1995. URL `ftp://megrez.math.u-bordeaux.fr/pub/pari`.

[2] Monagan, M. B. In-place arithmetic for polynomials over Z_n. In *Proceedings of DISCO '92* (1993), vol. 721 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 22–34.

[3] Monagan, M. B. von zur Gathen's factorization challenge. *SIGSAM Bulletin 27*, 2 (1993), 13–18.

[4] Shoup, V. A new polynomial factorization algorithm and its implementation. *Journal of Symbolic Computation 20*, 4 (Oct. 1995), 363–397.