Grand Challenges in Supercomputing at the Australian National University*

Terry Bossomaier, David Singleton, Margaret Kahn (editors)

Computer Sciences Laboratory and ANU Supercomputer Facility
Australian National University
Canberra, ACT 0200

April 1994

*Extract: only pages 34–39 are reproduced here (slightly edited).

rpb150 typeset using LaTeX

Integer Factorization

Richard P. Brent
Computer Sciences Laboratory

Summary

The problem of finding the prime factors of large composite numbers has always been of mathematical interest. With the advent of public key cryptosystems it is also of practical importance, because the security of some of these cryptosystems, such as the Rivest-Shamir-Adelman (RSA) system, depends on the difficulty of factoring the public keys.

In recent years the best known integer factorization algorithms have improved greatly, to the point where it is now easy to factor a 60-decimal digit number, and possible to factor numbers larger than 120 decimal digits, given the availability of enough computing power. However, the problem of integer factorization still appears difficult, both in a practical sense (for numbers of more than about 80 decimal digits), and in a theoretical sense (because none of the algorithms run in polynomial time).

We outline several recent integer factorization algorithms, including the elliptic curve algorithm (ECM), the multiple polynomial quadratic sieve (MPQS), and the special/general number field sieve (NFS), give examples of their use, and mention some applications.

Public key cryptography

Large primes have at least one practical application – they can be used to construct *public key* cryptosystems (also known as *asymmetric* cryptosystems and *open encryption key* cryptosystems) [12, 13]. The security of such systems depends on the (assumed) difficulty of factoring the product of two large primes. This is a practical motivation for the current interest in integer factorisation algorithms.

Parallel algorithms

We would hope that an algorithm which required time T_1 on a computer with one processor could be implemented to run in time $T_P \sim T_1/P$ on a computer with P independent processors. This is not always the case, since it may be impossible to use all P processors effectively. However, it is true for many integer factorisation algorithms, provided that P is not too large.

Integer factorization algorithms

There are many algorithms for finding a nontrivial factor f of a composite integer N. The most useful algorithms fall into one of two classes –

A. The run time depends mainly on the size of N, and is not strongly dependent on the size of f. Examples are –

- Lehman's algorithm [5], which has worst-case run time $O(N^{1/3})$.
- The Continued Fraction algorithm [9] and the Multiple Polynomial Quadratic Sieve algorithm [11], which under plausible assumptions have expected run time

  $$O(\exp(c(\log N \log\log N)^{1/2})),$$

  where c is a constant (depending on details of the algorithm).

- The (special) Number Field Sieve algorithm [6], which under plausible assumptions has expected run time

$$O(\exp(c(\log N)^{1/3}(\log\log N)^{2/3})),$$

where c is a constant, provided N has a suitable form.

B. The run time depends mainly on the size of f, the factor found. (We can assume that $f \le N^{1/2}$.) Examples are –

- The trial division algorithm, which has run time $O(f \cdot (\log N)^2)$.
- Pollard's "rho" algorithm [10], which under plausible assumptions has expected run time $O(f^{1/2} \cdot (\log N)^2)$.
- Lenstra's Elliptic Curve algorithm [8], which under plausible assumptions has expected run time

$$O(\exp(c(\log f \log\log f)^{1/2}) \cdot (\log N)^2),$$

where c is a constant.

In these examples, the time bounds are for a sequential machine, and the term $(\log N)^2$ is a generous allowance for the cost of performing arithmetic operations on numbers which are $O(N)$ or $O(N^2)$.

Due to space limitations we can not describe the algorithms in detail here. The interested reader is referred to the references given above and, for more details on implementations, [2, 3, 4, 14].

Examples

The *elliptic curve algorithm* (ECM) is the best known algorithm for finding moderately large factors of very large numbers. For example, ECM was used to complete the factorisation of the 617-decimal digit Fermat number $F_{11} = 2^{2^{11}} + 1$:

$$F_{11} = 319489 \cdot 974849 \cdot 167988556341760475137 \cdot 3560841906445833920513 \cdot p_{564}$$

where the 21-digit and 22-digit prime factors were found using ECM, and p_{564} is a 564-decimal digit prime. The factorisation required about 360 million multiplications mod N, which took less than 2 hours on a Fujitsu VP 100 vector processor [1, 2]. ECM is ideal for parallel implementation, because it involves many pseudo-random trials which may be performed in parallel on different processors, with little communication required.

The *multiple-polynomial quadratic sieve algorithm* (MPQS) is currently the best general-purpose algorithm for factoring moderately large numbers N whose factors are in the range $N^{1/3}$ to $N^{1/2}$. For example, A. K. Lenstra and M. S. Manasse recently found

$$\begin{aligned} 3^{329}+1 &= 2^2 \cdot 547 \cdot 16921 \cdot 256057 \cdot 36913801 \cdot 177140839 \cdot 1534179947851 \cdot \\ &\quad 24677078822840014266652779036768062918372697435241 \cdot p_{67}\ , \end{aligned}$$

where the penultimate factor is a 50-digit prime, and the largest factor p_{67} is a 67-digit prime. The computation used a network of workstations for "sieving", and a supercomputer for the final solution of a very large linear system.

Our numerical examples have involved numbers of the form

$$a^e \pm b,$$

for small a and b, although the ECM and MPQS factorisation algorithms do not take advantage of this special form. The (special) *number field sieve* (NFS) is a new algorithm which does take advantage of this special form. In concept it is similar to the quadratic sieve algorithm, but it works over an algebraic number field defined by a, e and b. We refer the interested reader to [6] for details, and merely give an example to show the power of the algorithm. Consider the 155-decimal digit Fermat number

$$F_9 = N = 2^{2^9} + 1.$$

Using NFS, Lenstra *et al* [7] found

$$F_9 = 2424833 \cdot 7455602825647884208337395736200454918783366342657 \cdot p_{99},$$

where p_{99} is an 99-digit prime. The 7-digit factor was already known, although this did not help much. The collection of relations took less than two months on a network of about 700 workstations. A sparse system with 226,688 rows and 199,203 columns was reduced to a dense matrix with about 72,413 rows and 72,213 columns. Using Gaussian elimination, dependencies (mod 2) between the rows were found in three hours on a 65,536-processor Connection Machine (CM2), and gave the desired factorisation of F_9.

There is currently much interest in extending the number field sieve algorithm to handle general numbers (i.e. not just numbers of a special form). This appears to be possible in theory, but it is not yet certain if all the practical difficulties can be overcome [6]. If they can, the (general) number field sieve algorithm should become the algorithm of choice for factoring integers larger than about 120 decimal digits. For smaller integers the MPQS algorithm will probably be faster (and certainly simpler) than NFS.

References

[1] R. P. Brent, "Factorisation of the eleventh Fermat number (preliminary report)", *AMS Abstracts* 10 (1989), 89T–11–73.

[2] R. P. Brent, "Parallel algorithms for integer factorisation", in *Number Theory and Cryptography* (edited by J. H. Loxton), Cambridge University Press, 1990. Preliminary version available by anonymous ftp from `nimbus.anu.edu.au:/pub/Brent/rpb115.dvi.Z`

[3] R. P. Brent, "Vector and parallel algorithms for integer factorisation", *Proc. Third Australian Supercomputer Conference*, Melbourne, 1990. Preliminary version available by anonymous ftp from `nimbus.anu.edu.au:/pub/Brent/rpb122.dvi.Z`

[4] T. R. Caron and R. D. Silverman, "Parallel implementation of the quadratic sieve", *J. Supercomputing* 1 (1988), 273–290.

[5] R. S. Lehman, "Factoring large integers", *Mathematics of Computation* 28 (1974), 637–646.

[6] A. K. Lenstra and H. W. Lenstra (editors), *The Development of the Number Field Sieve*, Lecture Notes in Mathematics 1554, Springer-Verlag, Berlin, 1993.

[7] A. K. Lenstra, H. W. Lenstra, Jr., M. S. Manasse and J. M. Pollard "The factorization of the ninth Fermat number", *Mathematics of Computation* 61 (1993), 319–349.

[8] H. W. Lenstra, "Factoring integers with elliptic curves", *Annals of Math.* (2) 126 (1987), 649–673.

[9] M. A. Morrison and J. Brillhart, "A method of factorisation and the factorisation of F_7", *Mathematics of Computation* 29 (1975), 183–205.

[10] J. M. Pollard, "A Monte Carlo method for factorisation", *BIT* 15 (1975), 331–334.

[11] C. Pomerance, J. W. Smith and R. Tuler, "A pipeline architecture for factoring large integers with the quadratic sieve algorithm", *SIAM J. on Computing* 17 (1988), 387–403.

[12] R. L. Rivest, A. Shamir and L. Adelman, "A method for obtaining digital dignatures and public-key cryptosystems", *Comm. ACM* 21 (1978), 120–126.

[13] J. Seberry and J. Pieprzyk, *Cryptography: An Introduction to Computer Security*, Prentice Hall, Sydney, 1989.

[14] R. D. Silverman, "The multiple polynomial quadratic sieve", *Mathematics of Computation* 48 (1987), 329–339.