5.1.3 The Number Field Sieve (NFS)

The number field sieve is currently supposed to be the asymptotically fastest known algorithm for factoring integers. It is conjectured to run in time

$$L_n\left[\frac{1}{3}, O(1)\right],$$

but this has not been rigorously proven.

The initial idea of the number field sieve is due to John Pollard (1988), who proposed it for the factorisation of a very special class of numbers; the Cunningham numbers of the form $r^e \pm s$, with r and s small. The modifications necessary to make it applicable to general N are due to Joe Buhler, Carl Pomerance, Hendrik W. Lenstra, Jr. and L. Adleman [59, 3]. Like the several variations of the quadratic sieve algorithms, the number field sieve attempts to factor N by solving the congruence

$$x^2 = y^2 \mod N,$$

subject to $x \neq \pm y \mod N$. Namely then $\gcd(x \pm y, N)$ is for each choice of the sign possibly a non-trivial divisor of N.

As explained for the quadratic sieve algorithms one looks for solutions

$$\text{“}square\text{”} \equiv \text{“}smooth\text{”} \mod N$$

instead of

$$\text{“}square\text{”} \equiv \text{“}square\text{”} \mod N.$$

Using linear algebra over GF (2), one can multiply many solutions of

$$\text{“}square\text{”} \equiv \text{“}smooth\text{”}$$

into one solution of “square ≡ square”. Also a solution of

$$\text{“}smooth\text{”} \equiv \text{“}smooth\text{”}$$

can be used (if x and y are smooth with $x = y \bmod N$, then $x^2 = xy \bmod N$ and xy is smooth as well).

The way factoring algorithms often generate smooth numbers is to generate small numbers and exploit the fact that these are more likely to be smooth than large ones. However, in any congruence $x = y \bmod N$ with $x \neq y$ at least one of x, y is larger of equal to $\frac{1}{2}N$ in absolute value, so that x, y cannot both be very small. But this problem can be circumvented by exploiting algebraic number fields.

For a suitable selection of the number field the degree of the number field d is chosen approximately to be

$$d = \left(\frac{3 \log N}{\log \log N}\right)^{\frac{1}{3}}.$$

This means in practice $d \approx 5$ or 6 for integers with 100 to 200 digits. Define $m := \lfloor N^{1/d} \rfloor$ and write N in base m:

$$N = \sum_{i=0}^{d} a_i m^i.$$

This implies $a_d = 1$, $0 \leq a_{d-1} \leq d$ and all other $a_i < m$. Define a polynomial:

$$f(X) = \sum_{i=0}^{d} a_i X^i \in Z[X].$$

This can be supposed to be irreducible, otherwise a factorisation of n is already found, which is very unlikely to happen.

Let α be a root of the polynomial f. Then the ring $\mathbb{Z}[\alpha]$ is considered being a subring of the number field $Q(\alpha)$. The algorithm works in the ring the elements of which can be thought of as vectors:

$$Z[\alpha] = Z \cdot 1 + Z \cdot \alpha + \ldots + Z \cdot \alpha^{d-1}.$$

The addition can be calculated componentwise and the multiplication has to be reduced modulo f.

The following isomorphism is important for the design of the algorithm: The ring $Z[\alpha]$ modulo the ideal $(\alpha - m)$, nothing more than evaluating the polynomials in the ring at m is isomorphic to the integers modulo N:

$$Z[\alpha]/(\alpha - m) \cong Z/NZ.$$

This isomorphism is used to collect equations of the type:

$$\text{“}small\text{”} = \text{“}small\text{”}$$

in the following way:

$$a + b\alpha = a + bm \bmod (\alpha - m).$$

The left hand side takes place in the abstract ring and the right hand side in the integers modulo N.

What is the meaning of "*small*" for the number field? It is determined by the size of the coefficients of the polynomial f and the vector size of the elements considered.

Out of this understanding of "*small*", the smoothness has to be considered, according to the factorisations by ideals in the abstract ring. This ring generally is not a unique factorisation domain, thus the known techniques from the quadratic sieves cannot be applied.

The representation of elements of $Z[\alpha]$ as sequences of exponents indexed by prime ideals is neither injective nor surjective. Especially they are not sufficient for the recognition of squares. But this problem can be solved by the use of quadratic characters ([3]). Thus for the NFS algorithm two different factor bases are needed, one for the abstract ring and one for the integers modulo N.

The number field sieve is not only theoretically a very interesting algorithm. It is remarkably fast in factoring very large integers of a special kind. For integers, that can be represented as a small polynomial with small coefficients, especially such of the form

$$r^e + s, \text{ with } r, s \text{ small},$$

it is substantially faster than ppmpqs. However, these are only a small

fraction of the entire set of integers and there is no method known for determining when such a representation is possible.

What can be said about general integers? Based on the analysis of norms that arise in the computation, it can be shown that the crossover point for general integers with the ppmpqs is somewhere between 140 and 150 digits. An implementation and running time analysis on a variety of numbers between 30 and 90 digits has been made. Extrapolation of this data confirms the theoretical crossover estimate. Thus it is not unlikely that the number field sieve is better than the ppmpqs for factoring numbers in the 512-bit range.

5.1.4 Exploiting the Power of Distributed Computing

The three factorisation algorithms mentioned above can be implemented straightforward on parallel computers. A very important point is, that one can distribute a lot of tasks to different processors without the need to receive all answers. Additionally it is very easy to prove the correctness of these processor answers.

This makes these algorithms well suited for distributed computing, allowing to collect idle-times of e.g. workstations not only of participants in a local area network but distributed over a wide network, possibly across the whole world. This was done with the factorisation of the ninth Fermat number and also for the 116 digit number