THE LATTICE SIEVE

J. M. POLLARD

SUMMARY. We describe a possible improvement to the Number Field Sieve. In theory we can reduce the time for the sieve stage by a factor comparable with $\log(B_1)$. In the real world, where much factoring takes place, the advantage will be less. We used the method to repeat the factorisation of F_7 on an 8-bit computer (yet again!).

OBJECT OF THE LATTICE SIEVE

We will consider the case of F_7 throughout. We have

$$2F_7 = x^3 + 2, \qquad \text{where } x = 2^{43}.$$

The object of the sieve stage of the NFS is to find pairs of small coprime integers a and b such that:

i. the integer $a + bx$ is smooth.

ii. the polynomial $N(a, b) = a^3 - 2b^3$ is smooth.

This polynomial arises as the 'norm' of the algebraic integer $a + bx$, where x is a root of the equation $x^3 + 2 = 0$. For the sieve stage we don't need to know anything about algebraic numbers (for the whole algorithm we seem to need to know a little—more for general numbers).

At present everyone assumes [1, 2, 3] that the best thing to do is fix b (positive) and sieve over a range of values of a (positive and negative). We propose a different method, suggested by the 'special q' version of the QS method (due to Davis and Holdridge).

We divide the factor base into two parts:

$$\begin{aligned} S&: \text{ the small primes:} && p \le B_0, \\ M&: \text{ the medium primes:} && B_0 < p \le B_1. \end{aligned}$$

We put $B_0/B_1 = k$, likely to be in the range 0.1–0.5; as usual, we also use:

$$L: \text{ the large primes:} \qquad B_1 < p \le B_2,$$

where B_2 is much larger than B_1.

Algorithm LS (the lattice sieve).

1. Choose a region R of the (a, b) plane to be sieved.
2. Choose a fixed prime q in M, the 'special prime', and sieve only those (a, b) pairs in R with

$$a + bx \equiv 0 \pmod{q}. \tag{1}$$

1991 *Mathematics Subject Classification.* Primary 11Y05, 11Y40.

Key words and phrases. Factoring algorithm, algebraic number fields.

The sieve is a double one, as usual:

i. We sieve the numbers $a + bx$ with the primes $p < q$ only,

ii. We sieve the numbers $N(a, b)$ with all the primes of S and M.

In both sieves we allow a large prime up to B_2. We discuss in a moment how the sieve works, but first we explain why it is (possibly) of interest.

Claim 1. The total number of integers sieved is much less than in the NFS.

In fact, it is reduced by the factor:

$$W = \sum_{q \in M} \frac{1}{q} \approx \frac{\log(1/k)}{\log(B_1)}. \tag{2}$$

Claim 2. We still get most of the solutions given by the NFS.

The ones we miss are those for which $a + bx$ has no prime factor in M. It is easiest to consider the case when no large prime is allowed in $(a + bx)$, that is, the *ff* and *fp* solutions of [1, 2]; then the fraction of solutions lost, L say, can be expressed in terms of Dickman's ρ function [4]. A random integer of size about bx has all factors $\leq B_1$ with probability:

$$\rho(\ln(bx)/\ln(B_1)), \tag{3}$$

and all factors $\leq B_0$ with probability:

$$\rho(\ln(bx)/\ln(B_0)). \tag{4}$$

The required fraction L is (4)/(3).

Example. The factorisation of F_9 [2].

Here we have $x = 2^{103}$, $b = 1.25(6)$ [middle of the range], $B_1 = 1.3(6)$. The table below compares:

W = work done (eqn. 2), and

L = fraction of solutions lost,

as functions of k.

k	W	$\ln(bx)/\ln(B_0)$	(4)	$L = (4)/(3)$
1.0	0.0	6.0687	1.71(−5)	1.0
0.5	0.0492	6.3830	6.38(−6)	0.373
0.4	0.0651	6.4912	4.52(−6)	0.264
0.3	0.0855	6.6363	2.85(−6)	0.167
0.2	0.1143	6.8521	1.42(−6)	0.083
0.1	0.1636	7.2554	3.81(−7)	0.022

Notes.

1. We used an approximate formula for ρ given by Pomerance [4] (ln is the natural logarithm):

$$\rho(u) = \exp(-u(\ln(u) + 0.56 - 1/\ln(u))), \qquad (5 < u < 11).$$

2. It may be that a larger proportion of the *pf* and *pp* solutions are lost.

Conclusion. An Infinitely Skilful Programmer (ISP) can get 83% of the solutions for 8.6% of the work.