Predicting the Sieving Effort for the Number Field Sieve

Willemien Ekkelkamp[1,2]

[1] CWI, P.O. Box 94079, 1090 GB Amsterdam, The Netherlands
[2] Leiden University, P.O. Box 9512, 2300 RA Leiden, The Netherlands
W.H.Ekkelkamp@cwi.nl

Abstract. We present a new method for predicting the sieving effort for the number field sieve (NFS) in practice. This method takes relations from a short sieving test as input and simulates relations according to this test. After removing singletons, we decide how many relations we need for the factorization according to the simulation and this gives a good estimate for the real sieving. Experiments show that our estimate is within 2 % of the real data.

1 Introduction

One of the most popular methods for factoring large numbers is the number field sieve [4], as this is the fastest algorithm known so far. In order to estimate the most time-consuming step of this method, namely the sieving step in which the so-called relations are generated, one looks at actual sieving times for numbers of comparable size. If these are not available, one could try to extrapolate actual sieving times for smaller numbers, using the formula for the running time $L(N)$ of this method, where N is the number to be factored. We have

$$L(N) = \exp(((64/9)^{1/3} + o(1))(\log N)^{1/3}(\log\log N)^{2/3}), \text{ as } N \to \infty \ ,$$

where the logarithms are natural. These estimates can be 10–30 % off.

In this paper we present a method for predicting the number of relations needed for factoring a given number in practice within 2 % of the actual number of relations needed. With 'in practice' we mean: on a given computer, for a given implementation, and for a given choice of the parameters in the NFS. This allows us to predict the actually required sieving time within 2 %. Our method is based on a short sieving test and a very cheap simulation of the relations needed for the factorization. By applying this method for various choices of the parameters of the number field sieve, it is possible to find an optimal choice of the parameters, e.g., in terms of minimal sieving time or in terms of minimizing the size of the resulting matrix. Before going into details we give a short overview of the NFS in order to show where our method fits in.

The NFS consists of the following four steps. First we select two irreducible polynomials $f_1(x)$ and $f_2(x)$, $f_1, f_2 \in \mathbb{Z}[x]$, and an integer $m < N$, such that

$$f_1(m) \equiv f_2(m) \equiv 0 \pmod{N} \ .$$

Polynomials with 'small' integer coefficients are preferred, because the values of these polynomials are smaller on average and smoother (i.e. having smaller prime factors on average) than the values of polynomials with large integer coefficients. Usually $f_1(x)$ is a linear polynomial and $f_2(x)$ a higher degree polynomial, referred to as rational side and algebraic side, respectively. If N is of a special form (e.g., $c^n \pm 1$) then we can use this to get a polynomial $f_2(x)$ with very small coefficients. In that case we talk about the special number field sieve (SNFS), else we talk about the general number field sieve (GNFS). By α_1 and α_2 we denote roots of $f_1(x)$ and $f_2(x)$, respectively.

The second step is the relation collection. We choose a factorbase FB of primes below the bound F and a large primes bound L; for ease of exposition we take the same bounds on both the rational side and the algebraic side. Then we search for pairs (a, b) such that $\gcd(a, b) = 1$, and such that both $F_1(a, b) = b^{\deg(f_1)} f_1(a/b)$ and $F_2(a, b) = b^{\deg(f_2)} f_2(a/b)$ have all their prime factors below F and at most two prime factors between F and L, the so-called large primes. These pairs (a, b) are referred to below as relations (a_i, b_i).

There are many possibilities for the relation collection, the fastest of which are based on sieving. Two sieving methods in particular are widely used, namely line sieving and lattice sieving. For line sieving we select a rectangular sieve area of points (a, b) and the sieving is done per horizontal line. For lattice sieving we select an interval of so-called special primes and for each special prime we only sieve those pairs (a, b) for which this special prime divides $b^{\deg(f_2)} f_2(a/b)$; for each special prime these pairs form a lattice in the sieving area. In case of SNFS the special prime is chosen on the rational side.

The third step consists of linear algebra to construct a set S of indices i such that the two products $\prod_{i \in S}(a_i - b_i\alpha_1)$ and $\prod_{i \in S}(a_i - b_i\alpha_2)$ are both squares of products of prime ideals. This product comes from the fact that $b^{\deg(f_1)} f_1(a/b)$ is the norm of the algebraic number $a - b\alpha_1$, multiplied with the leading coefficient of $f_1(x)$. The principal ideal $(a - b\alpha_1)$ factors into the product of prime ideals in the number field $\mathbb{Q}(\alpha_1)$. The situation is similar for f_2.

The last step is the square root step. We determine algebraic numbers $\alpha_1' \in \mathbb{Q}(\alpha_1)$ and $\alpha_2' \in \mathbb{Q}(\alpha_2)$ such that $(\alpha_1')^2 = \prod_{i \in S}(a_i - b_i\alpha_1)$ and $(\alpha_2')^2 = \prod_{i \in S}(a_i - b_i\alpha_2)$. Then we use the homomorphisms $\phi_{\alpha_1} : \mathbb{Q}(\alpha_1) \to \mathbb{Z}/N\mathbb{Z}$ and $\phi_{\alpha_2} : \mathbb{Q}(\alpha_2) \to \mathbb{Z}/N\mathbb{Z}$ with $\phi_{\alpha_1}(\alpha_1) = \phi_{\alpha_2}(\alpha_2) = m$ to get $\phi_{\alpha_1}(\alpha_1')^2 = \phi_{\alpha_1}\left((\alpha_1')^2\right) = \phi_{\alpha_1}\left(\prod_{i \in S}(a_i - b_i\alpha_1)\right) \equiv \prod_{i \in S}((a_i - b_i m) \equiv \phi_{\alpha_2}(\alpha_2')^2 (\mathrm{mod}\ N)$. Now compute $\gcd(\phi_{\alpha_1}(\alpha_1') - \phi_{\alpha_2}(\alpha_2'), N)$ to obtain a factor of N. If this gives the trivial factorization, continue with the next set of indices, otherwise we have found a nontrivial factorization of N. For more details of the NFS, see e.g., [3], [4], or [5].

Our method works as follows. After choosing polynomials, bounds F and L, and a sieve area, we perform a sieve test for a relatively short period of time. For a 120-digit N one could sieve for ten minutes or so, but for larger numbers one may spend considerably more time on the sieve test. Based on the relations in this sieve test we simulate as many relations as are necessary for factoring the number. The simulation uses a random number generator and functions that

describe the underlying distribution of the large primes, and this can be done fast. During the simulation of the relations, we regularly remove the singletons from all the relations simulated so far. As soon as the number of relations left after singleton removal exceeds the number of primes in the relations we stop and it turns out that the total number of relations simulated so far gives us a good estimate of the actual number of relations that we need to factor our number.

The number of useful relations after singleton removal grows in a hard-to-predict fashion as a function of the number of relations found. This growth behaviour differs from number to number, which makes it hard to predict the overall sieving time: for instance, even estimates based on factoring times of numbers of comparable size can easily be 10 % off. Our method, however, which is purely based on the individual behavior of the relations found for the number to be factored, allows us to predict how the number of useful relations will behave as a function of the number of relations found, thereby giving us a tool to accurately predict the overall sieving time.

The simulations in this paper were carried out on a Intel® Core™2 Duo with 2 GB of memory. The line sieving data sets were generated with the NFS software package of CWI. The lattice sieving data sets were given by Bruce Dodson and Thorsten Kleinjung.

In Section 2 we describe how we simulate the relations. Section 3 is about the singleton removal and about how to decide when we have enough relations to factor the given number. In Section 4 we compare results of the simulation with real factorizations and Section 5 contains the conclusions and our intentions for future work.

2 Simulating Relations

Before we start with the simulation, we run a short sieving test. In order to get a representative selection of the actual relations, we ensure that the points we are sieving in this test are spread over the entire sieving area. The parameters for the sieving are set in such a way that we have at most two large primes both on the rational side and on the algebraic side. In the case of lattice sieving we have one additional special prime on one of the sides. In this section we describe the process of simulating relations both for line sieving and for lattice sieving. Note that we only simulate the large primes; for the primes in the factorbase we use a correction as will be explained in Section 3.

The first step after the sieving test consists of splitting the relations according to the number of large primes. The set of relations with i large primes on the rational side and j large primes on the algebraic side is denoted by $r_i a_j$ for $i, j \in \{0, 1, 2\}$. This leads to nine different sets and the mutual ratios of their cardinalities determine the ratios by which we will simulate the relations. In the case of lattice sieving we split the relations in the same way, ignoring the special prime.

Next we take a closer look at the relations in each set and specify a model

that fits the distribution of the large primes in these sets as closely as we can accomplish. To clarify this, we explain for each set how to simulate the relations in that set, for the case of line sieving.
r_0a_0: We count the number of relations in this set.
r_1a_0: We started with sorting all the large primes and put them in an array. Our first experiments with simulating the large primes (and removing singletons) concentrated on the large primes at hand. We tried linear interpolation between two consecutive large primes, Lagrange polynomials, and splines, but all these local approaches did not give a satisfying result; the result after singleton removal was too far from the real data. We then tried a more global approach, looking at all the large primes and see if we could find a distribution for them. We found that an exponential distribution simulates best the distribution of these large primes over the interval $[F, L]$ (cf. [2], Ch. 6) and the result after singleton removal was satisfactory. The inverse of this distribution function is given by

$$G(x) = F - a \log\left(1 - x\left(1 - e^{\frac{F-L}{a}}\right)\right), 0 \leq x \leq 1 \ , \tag{1}$$

where a is the average of the large primes in the set r_1a_0. Note that $G(0) = F$ and $G(1) = L$. In order to generate primes according to the actual distribution of the large primes, we generate a random number between 0 and 1, substitute this number in $G(x)$, round the number $G(x)$ to the nearest prime, and repeat this for each prime that we want to generate.

To avoid expensive prime tests, we work with the index of the primes p, defined as $i_p = \pi(p)$, rather than with the prime itself. This index can be found by using a look-up table or the approximation $i_p \approx \frac{p}{\log p} + \frac{p}{\log^2 p} + \frac{2p}{\log^3 p}$ [6]. Experiments showed that this third order approximation gives almost the same results as looking up indices in a table. It is especially more efficient to use this approximation when L is large. For working with indices, we have to adjust (1); we write i_F for the index of the first prime above F, and i_L for the index of the prime just below L, and a' for the average of the indices of the large primes in the set r_1a_0. Then the formula becomes

$$G(x) = i_F - a' \log\left(1 - x\left(1 - e^{\frac{i_F - i_L}{a'}}\right)\right) \ . \tag{2}$$

To illustrate that the distribution of the large primes is approximated well by (2) we have generated the following graph, which consists of two sorted sets. One set consists of the indices of the primes of the original sieving data and the other set consists of the indices simulated with help of (2). The line of the simulated data is the one which lies below the other line (of the original data) around position 7000.

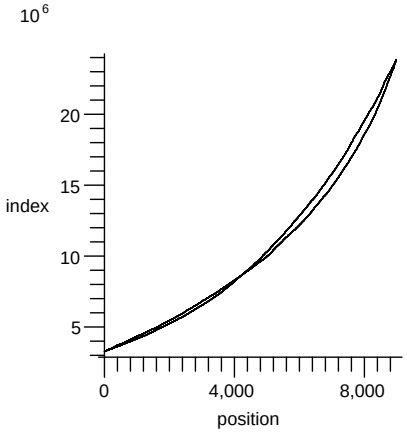

Fig. 1. Comparing original and simulated data

The necessary number of relations in the set r_1a_0 depends on how many relations we have to generate in total.
$\underline{r_0a_1}$: We would like to use the same idea as we used for r_1a_0, but now we have to deal with algebraic primes. This means that not all primes can occur, and that each prime that does occur can have up to d different roots, where d is the degree of the polynomial $f_2(x)$. This yields pairs of a prime and a root which we denote by $(prime, root)$. Luckily, (heuristically) the amount of pairs $(prime, root)$ with $F < prime < L$ is about equal to the amount of primes between F and L. This implies that we do not have to simulate pairs with a certain subset of indices, as we may assume that all indices can occur in the simulation. We found that an exponential distribution fits here as well, so here we use the same approach as we did for r_1a_0.
$\underline{r_1a_1}$: We know now how to simulate r_1a_0 and r_0a_1, and we assume that the value of the index on the rational side is independent of the value of the index on the algebraic side. We combine both approaches: using (2), generate a random number and compute the corresponding rational index, generate a new random number (do not use the first random number as input for the random number generator) and compute the algebraic index.
$\underline{r_2a_0}$: Here we have to deal with two large primes on the rational side, denoted by q_1 and q_2 with $q_1 > q_2$. We started with sorting the list with q_1 and (to our surprise) we found that a linear distribution fits these data well. So the distribution function of the index i_{q_1} of q_1 is given by

$$H_1(x) = i_F + x(i_L - i_F) \ ,$$

where x is a number between 0 and 1.

We continued with q_2 and sorted them. Here, an exponential distribution fits the data, but now we have to take into account that $q_2 < q_1$. Remember that we need an average value for the exponential distribution, but we cannot use all q_2-values. Instead of using one average value, we make a list of averages a_{q_2} of the sorted q_2-indices, where $a_{q_2}[j]$ contains the average of the first j q_2-indices.

Now we describe how to simulate elements of r_2a_0. We begin with a random number between 0 and 1 and compute $H_1(x)$, which gives us an index i_{q_1} of q_1. We look up this index in the sorted list of q_2-indices and the corresponding position j tells us which average we should use for computing the index i_{q_2} of q_2. We generate a new random number between 0 and 1 and substitute it for x in the following formula $H_2(x)$, which is an adjusted form of $G(x)$:

$$H_2(x) = i_F - a_{q_2}[j] \log\left(1 - x\left(1 - e^{\frac{i_F - i_{q_1}}{a_{q_2}[j]}}\right)\right) \quad .$$

This gives us an index i_{q_2} of q_2 that is smaller than the index we generated for q_1.

Our observation of a linear distribution of the largest prime and an exponential distribution of the second prime may not be as one would expect theoretically, but this might very well be a consequence of sieving in practice. For example, products of size approximately L^2 factor most of the time as one prime below L and one prime above L and are discarded. Thus most sievers do not spend much time on factors of this size. It may turn out to be the case that a siever with different implementation choices gives rise to different distributions, which needs to be investigated further.

To illustrate the distribution of the products of the two large primes for the dataset of $13,220+$ (cf. Section 4) found by our implementation of the siever, we added for each relation in r_2a_0 the indices of the two large primes and split the interval $[2i_F, 2i_L]$ in ten equal subintervals (labeled $s = 1, \ldots, 10$). For each subinterval we counted the number of relations for which the sum of the two indices of the two large primes lies in this subinterval: see Table 1.

Table 1. Distribution of the sum of the indices $(13,220+)$

s	1	2	3	4	5	6	7	8	9	10
# relations	120780	161735	148757	133845	121967	78725	39253	20710	8107	0

The zero in the last column is due to one of the bounds in the siever, which was set at $F^{0.1}L^{1.9}$ instead of L^2.

$\underline{r_0a_2}$: We know how to deal with r_2a_0 and we apply the same approach to r_0a_2, as we can make the same transition as we made from r_1a_0 to r_0a_1.

Sorting the list with q_1 showed that we could indeed use a linear distribution and the sorted list with q_2 showed that an exponential distribution fitted here.

Now we simulate elements of r_2a_0 in the same way as elements of r_0a_2.
$\underline{r_1a_2}$: As with r_1a_1, we assume that the rational side and the algebraic side are independent. Here we combine the approaches of r_1a_0 and r_0a_2 to get the elements of r_1a_2.
$\underline{r_2a_1}$: Combine the approaches of r_2a_0 and r_0a_1 to get the elements of r_2a_1.
$\underline{r_2a_2}$: As in the previous two sets, we combine two approaches, this time r_2a_0 and r_0a_2.

Summarizing, our simulation model consists of four assumptions:

- the rational side and the algebraic side are independent,
- the rational side and the algebraic side are equivalent,
- a model for one large prime (described in r_1a_0),
- a model for two large primes (described in r_2a_0).

In case of lattice sieving, we simulate the relations in the same way and add a special prime to all the relations in the following way. We compute the average number of relations per pair $(special\ prime, root)$ in the sieving test. Then we divide the number of relations we want to simulate by this average and this gives the total number of special primes in our simulation. Then we select an appropriate interval from which the special primes are chosen. Divide this interval in a (small) number of sections: per section select randomly the special primes and add each of these special primes to a relation. By dividing in sections (and simulating the same amount of relations per section) we make sure that the entire interval of special primes is covered, but by choosing randomly in each section, we get enough variation in the amount of relations per special prime. If the interval of the special primes is very large, it might become necessary to decrease the number of relations per section. In our example this was not the case, but a well-chosen sieve test will give this information.

It is possible to use different factorbase bounds for the rational primes and the algebraic primes, bound the product of the two large primes on the same side, etc. All these details in the sieving influence the relations, but once the general model is known, it is relatively easy to adjust it to match the details.

3 The Stop Criterion

We now know how to simulate relations, but how many should we simulate?

In order to factor the number N we have to find dependencies in a matrix, which is determined by the relations, as mentioned in the introduction in the third step of the NFS. Every column is identified with a prime $\leq L$ (rational and algebraic primes). Suppose each row represents a relation. If a prime occurs an odd number of times in that relation, we put a one in the column of that prime and a zero otherwise. After representing all relations in this matrix, we remove those relations with a 1 that is the only 1 in the entire column, the so-called singletons. This may generate new singletons, so this singleton removal step is repeated until all primes occur at least twice. In practice, this is done before

actually building a matrix.

For our stop criterion it is enough to know when we have enough relations, i.e. *when the number of relations after singleton removal exceeds the number of different primes that occur in the remaining relations.*

After the singleton removal, we count how many relations are left and how many different large primes occur in these relations. We define the percentage oversquareness O_r after singleton removal (s.r.) as

$$O_r := \frac{n_r}{n_l + n_F - n_f} \times 100 \ ,$$

where n_r is the number of relations after singleton removal, n_l is the number of different large primes after singleton removal, n_F is the number of primes in the factorbase, approximated by $\pi(F_{\text{rat}}) + \pi(F_{\text{alg}})$, and n_f is the number of free relations from factorbase elements. We have ([3], Ch. 3):

$$n_f = \frac{1}{g}\pi(\min(F_{\text{rat}}, F_{\text{alg}})) \ ,$$

where g is the order of the Galois group of $f_1(x)f_2(x)$. If $O_r \geq 100\,\%$ we may expect to find a dependency in the matrix, and we may stop with simulating relations. To make practically sure to find a dependency, we may stop at $102\,\%$. Even a larger percentage is allowed if one would like to have more choice in the relations that can form a dependency and subsequently form a smaller matrix in the linear algebra step.

One final point concerns lattice sieving. It is well known that lattice sieving produces lots of duplicates, especially when it involves many special primes. We treat our relations as if there are no duplicates, but that implies that in the case of lattice sieving we have to add a certain number of relations to the relations that we should collect in the sieving stage. This number can be computed as in [1]. The basic idea in [1] is to run a small sieve test and find out which relations have more than one prime in the special primes interval. If such a relation would be found by more than one lattice in the sieving area (remember that each special prime gives rise to a lattice in the sieving area), than this gives a duplicate relation.

4 Experiments

We have applied our method to several real data sets (coming from factored numbers) and show that this gives good results. We have carried out two types of experiments.

First we assumed that the complete data set is given and we wanted to know if the simulation gave the same oversquareness when simulating the same number of relations as is contained in the original data set. For the simulation we used $0.1\,\%$ of the original data.

Secondly we assumed that only a small percentage ($0.1\,\%$) of the original

data is known. Based on this data we simulated relations until $O_r \geq 100\,\%$. Then we compared this with the oversquareness of the same number of original relations.

This 0.1 % is somewhat arbitrary. We came to it in the following way: we started a simulation based on 100 % real data and lowered this percentage in the next experiment until results after singleton removal were too far from the real data. We went down as far as 0.01 %, but this percentage did not always give good results, unless we would have been satisfied with an estimate within 5 % of the real data (although some experiments with 0.01 % of the real data were even as good as the ones based on 0.1 % of the real data).

4.1 Line Sieving

Some relevant parameters for all the real data sets in this section are given in Table 2, where M stands for million. Numbers are written in the format $a, b+$ or $a, b-$, meaning $a^b + 1$ or $a^b - 1$. In the case of GNFS, some prime factors were already known and for the remaining factors it was more efficient to use GNFS instead of SNFS.

Table 2. Sieving parameters (line sieving)

number	# dec. digits	F	L	g	$n_F - n_f$
13,220+	117	30M	400M	120	3700941
26,142+	124	30M	250M	120	3700941
19,183−	131	30M	250M	18	3613192
66,129+	136	35M	300M	18	4175312
80,123−	150	55M	450M	18	6383294

The experiments for the first two GNFS data sets $13,220+$ and $26,142+$ are in Table 3. Here, O stands for the original data and S for the simulated data. Table 3 shows that the numbers were oversieved, but the simulated data show about the same oversquareness. In Table 4, we computed the relative difference $(\mathrm{S}-\mathrm{O})/\mathrm{O} \times 100\,\%$ of the entries in the S- and O-column of Table 3. We see that our predictions of the number of relations after s.r., the number of large primes after s.r., and the oversquareness are close to the real data to about 1 %.

Table 3. Experiments line sieving

GNFS	13,220+ O	13,220+ S	26,142+ O	26,142+ S
# relations before s.r.	35 496 483	35 496 483	23 580 294	23 580 294
# relations after s.r.	21 320 864	21 394 640	15 150 790	15 253 825
# large primes after s.r.	13 781 518	13 950 420	9 448 082	9 397 751
oversquareness (%)	121.96	121.21	115.22	116.45

Table 4. Relative differences of Table 3 results

GNFS	13,220+	26,142+
relations after s.r. (%)	0.35	0.68
large primes after s.r. (%)	1.22	−0.53
oversquareness (%)	−0.61	1.07

We give the following timings for these experiments: simulation of the relations, singleton removal, and real sieving time (Table 5). For the actual sieving we used multiple machines and added the sieving times of each machine. As we used 0.1 % data, we have to keep in mind that we need to add 0.1 % of the sieving time to a complete experiment, which consists of generating a small data set, simulate a big data set, and remove singletons. When we change parameters in the NFS we have to generate a new data set.

Roughly speaking, we can say that one prediction of the total sieving time (for a given choice of the NFS parameters) with our method costs less than one CPU hour, whereas the actual sieving costs several hundreds of CPU hours.

Table 5. Timings

GNFS	13,220+	26,142+
simulation (sec.)	224	156
singleton removal (sec.)	927	573
actual sieving (hrs.)	316	709

Now for our second type of experiments, we assume that we only have a small sieve test of the number to be factored. When are we in the neighbourhood of 100 % oversquareness according to our simulation and will the real data agree with our simulation? We started to simulate 5M, 10M, ... relations (with increment 5M) and for these numbers we computed the oversquareness O_r; when O_r approached the 100 % bound we decreased the increment to 1M. Table 6 gives

the number of relations for which O_r is closest to 100 % and the next O_r (for 1M more relations), both for the simulated data and the original data. This may of course be refined.

Table 6. Around 100 % oversquareness (GNFS)

# rel. before s.r.	O_r S (%)	O_r O (%)	rel. diff. (%)
28M (13,220+)	99.66	99.87	−0.21
29M (13,220+)	103.15	103.29	−0.14
20M (26,142+)	100.57	99.24	1.34
21M (26,142+)	105.38	104.03	1.30

For SNFS the higher degree polynomial has small coefficients. Tables 7–10 show the same kind of data as Tables 3–6, but now for SNFS. We start in Table 7 with the complete data set and simulate the same number of relations. Table 8 gives the relative differences of the results of the experiments in Table 7. The timings are given in Table 9.

Table 7. Experiments line sieving

SNFS	# rel. before s.r.	# rel. after s.r.	# l.p. after s.r.	oversquareness (%)
19,183− O	21 259 569	11 887 312	7 849 531	103.70
19,183− S	21 259 569	12 156 537	7 936 726	105.25
66,129+ O	26 226 688	15 377 495	10 036 942	108.20
66,129+ S	26 226 688	15 656 253	10 123 695	109.49
80,123− O	36 552 655	20 288 292	12 810 641	105.70
80,123− S	36 552 655	20 648 909	12 973 952	106.67

Table 8. Relative differences of Table 7 results

SNFS	19,183−	66,129+	80,123−
relations after s.r. (%)	2.26	1.81	1.78
large primes after s.r. (%)	1.11	0.86	1.27
oversquareness (%)	1.49	1.19	0.92

Table 9. Timings

SNFS	19,183−	66,129+	80,123−
simulation (sec.)	128	166	223
singleton removal (sec.)	487	603	771
sieving (hrs.)	154	197	200

In Table 10 we simulate the number of relations that leads to an oversquareness around 100 %. We compare this number with the real data and give the differences in oversquareness.

Table 10. Around 100 % oversquareness (SNFS)

# rel. before s.r.	O_r S (%)	O_r O (%)	rel. diff. (%)
20M (19,183−)	99.22	97.71	1.55
21M (19,183−)	104.06	102.51	1.51
23M (66,129+)	96.44	95.35	1.14
24M (66,129+)	100.72	99.60	1.12
34M (80,123−)	99.93	98.66	1.29
35M (80,123−)	102.82	101.50	1.30

All these data sets were generated with the NFS software package of CWI, and the models for describing the underlying distributions were the same for SNFS and GNFS, as described in Section 2.

4.2 Lattice Sieving

For lattice sieving we used a data set from Bruce Dodson (7,333−, SNFS). Besides the factorbase bound and the large primes bound, we have two intervals for the special primes. These are given in Table 11.

Table 11. Sieving parameters (lattice sieving)

	7,333−
# dec. digits	177
F	16 777 215
L	250 000 000
special primes	[16 777 333, 29120617]
	[60 000 013, 73 747 441]
g	6
$n_F - n_f$	1 976 740

As we are now dealing with lattice sieving, we have an extra (special) prime to simulate, in the way described in Section 2. Fortunately, the distribution of the other large primes did not change. The results of our experiments are given in Table 12, based on 0.023 % original data. The last line in this table is the total number of relations without duplicates. In total 26 024 921 relations were sieved.

Table 12. Oversquareness 7,333−

# rel. before s.r.	O_r S (%)	O_r O (%)	rel. diff. (%)
17M (7,333−)	98.34	97.45	0.91
18M (7,333−)	103.96	103.08	0.85
25 112 543 (7,333−)	135.39	136.64	−0.91

Apart from receiving a lattice sieving data set from Bruce Dodson, we also received lattice sieving data sets from Thorsten Kleinjung. Unfortunately the model described in this paper for the large primes does not yield satisfactory results for the latter data sets.

5 Conclusions and Future Work

Our experiments show that our simulation of the relations works well. Based on a small fraction of the sieving data, we obtain a good model of the distribution of the large primes in the relations. Combined with singleton removal, our estimation of the oversquareness is within 2 % of the real data. Thus we cheaply obtain a good estimate of the number of necessary relations for factoring a given number on a given computer, and hence of the actual computing time. Therefore, this method is a useful tool for optimizing parameters in the number field sieve, and we actually are using it in our practical factorization work.

Future work will include finding the correct model for the lattice sieve data sets of Kleinjung and check to which extent this model depends on the implementation of the siever. A second objective is to find a theoretical explanation for the occurrence of the various distributions (linear, exponential, ...) of the large primes. Another objective will be to find the optimal oversquareness for minimizing the resulting matrix. Once these issues are properly understood we intend to develop a tool to determine bounds F and L that optimize the overall effort for relation collection and matrix processing with respect to the available resources.

Acknowledgements

The author thanks Arjen Lenstra for suggesting the idea to predict the sieving effort by simulating relations on the basis of a short sieving test. She thanks

Marie-Colette van Lieshout for suggesting several statistical models including the model which is used in Section 2, $r_1 a_0$, and Dag Arne Osvik for providing the singleton removal code for relations written in a special format.

The author thanks Arjen Lenstra, Herman te Riele, and Rob Tijdeman for reading the paper and giving constructive criticism and comments, Bruce Dodson and Thorsten Kleinjung for sharing data sets, and the anonymous referees for carefully reading the paper and suggesting clarifications.

Part of this research was carried out while the author was visiting École Polytechnique Fédérale de Lausanne in August 2006. She thanks Arjen Lenstra and EPFL for the hospitality during this visit.

References

1. K. Aoki, J. Franke, T. Kleinjung, A.K. Lenstra, D.A. Osvik: A kilobit special number field sieve factorization. Adv. Crypt. - ASIACRYPT 2007, LNCS 4833 (2007) 1–12
2. Breiman, L.: Statistics: With a View Toward Applications. Houghton Mifflin Company Boston, 1973
3. Elkenbracht-Huizing, M.: The Number Field Sieve. Ph.D. thesis, University of Leiden, 1997
4. Lenstra, A.K., Lenstra, H.W., Jr. (Eds.): The Development of the Number Field Sieve. Lecture Notes in Math., vol. 1554, Springer-Verlag, Berlin, 1993
5. Montgomery, P.L.: A survey of modern integer factorization algorithms. CWI Quarterly, **7/4** (1994) 337–366
6. Panaitopel, L.: A formula for $\pi(x)$ applied to a result of Koninck-Ivić. Nieuw Arch. Wiskunde, **5/1** (2000) 55–56