# CONTINUED FRACTIONS AND LATTICE SIEVING

JENS FRANKE, THORSTEN KLEINJUNG

Abstract. We present a new method of lattice sieving which we expect to be faster by a constant factor than the method of Pollard, and which has been used in recent GNFS records. We also explain how to efficiently split the sieving region among several computing nodes and analyze the asymptotic behaviour of the cost of sieving on a large parallel computer.

The asymptotic behaviour of the cost parallelized sieving has recently been analyzed by D. Bernstein ([Ber]), who assumed that a two-dimensional mesh is used. We propose a parallelized lattice siever using a butterfly-like topology. The Bernstein cost function for this siever is superior to the cost function for the methods proposed by Bernstein, both asymptotically and for projects of a size comparable to current factorization records. For very large projects, of a size well above RSA1024, one may encounter problems realizing this topology in three-dimensional Euclidean space. We will explain in Remark 3 in the last section that this problem is unlikely to occur for projects of a feasible size.

## 1. The algorithm for lattice sieving

We start with an outline of the procedure of lattice sieving, which was introduced by Pollard. We give a brief description of the problems encountered when one wants to implement this in an efficient way. These problems, as well as a practical solution to them, were also described by Pollard. We present a solution which we believe to be more elegant and which probably saves a constant factor over Pollard's method.

A special $q$-pair is a pair $(q, \rho)$ such that $q$ is a prime and $f(a, b) \equiv 0 \bmod p$ if $a \equiv \rho b \bmod q$. To carry out lattice sieving for a special $q$ pair, one calculates a reduced basis $\big((a_0, b_0), (a_1, b_1)\big)$ of the lattice of all $(a, b) \in \mathbb{Z}^2$ with $a \equiv \rho b \bmod q$. For highly skewed GNFS polynomials, a weighted scalar product is used for the reduction. Then one looks for pairs $(a, b) = i(a_0, b_0)+j(a_1, b_1)$, with $(i, j)$ belonging to the sieving region $\mathcal{A}$ defined by the two conditions $-I/2 \le i < I/2$ and $0 < j \le J$, for which $f(a, b)/q$ is $B_0$-smooth and $g(a, b)$ is $B_1$-smooth, where $B_0$ and $B_1$ are appropriate factor base bounds for the lattice siever. One is only interested in such a pair if $i$ and $j$ are coprime. To determine such pairs, one expresses both polynomials in $(i, j)$-coordinates

$$h_0(i, j) = \frac{f(ia_0 + ja_1, ib_0 + jb_1)}{q} \qquad h_1(i, j) = g(ia_0 + ja_1, ib_0 + jb_1)$$

and considers the factor bases $\mathfrak{B}_\lambda$ consisting of all sublattices $\mathfrak{p} \subset \mathbb{Z}^2$ such that the order $\mathbb{Z}^2/\mathfrak{p}$ is a prime $p =: N(\mathfrak{p}) < B_\lambda$ which divides $h_\lambda(i, j)$ if $(i, j) \in \mathfrak{p}$. Note that, with finitely many exceptions, $N(\mathfrak{p})$ is the norm of the ideal belonging to $\mathfrak{p}$. For each $\lambda \in \{0; 1\}$, one calculates $\mathfrak{B}_\lambda$ (by expressing a precalculated factor base for the original polynomial pair in $(i, j)$-coordinates), initializes a sieve array with $S[i][j] \leftarrow 0$ for all $(i, j) \in \mathcal{A}$, and then adds $\log(N(\mathfrak{p}))$ to $S[i][j]$ for each $\mathfrak{p} \in \mathcal{B}_\lambda$ and each $(i, j) \in \mathfrak{p} \cap \mathcal{A}$. Then one takes a closer look at the $(i, j) \in \mathcal{A}$ for which $S[i][j]$

is large after both passes. All implementations we are aware of use a sieve array of bytes and a one byte approximation to $\log(N(\mathfrak{p}))$. This appears to be preferable since memory throughput is one of the limiting factors for the speed of sieving, and since higher accuracy is not normally necessary.

If $p = N(\mathfrak{p})$ does not exceed a small multiple of $I$ and if $\mathfrak{p}$ does not correspond to the infinity point of the projective line $\mathbb{P}^1(\mathbb{Z}/p\mathbb{Z})$, then sieving proceeds by the usual line sieving procedure as follows: For $0 < j \leq J$, one attempts to calculate the smallest $i$ with $(i, j) \in \mathcal{A} \cap \mathfrak{p}$, proceeding to the next factor base element if no such $i$ exists. Then one increments $S[i][j]$ by $\log(N(\mathfrak{p}))$ and, replacing $i$ by $i + p$, iterates this procedure until $i \geq I/2$. If $\mathfrak{p}$ is defined by the condition $p|j$, then the same procedure may be used if the roles of $i$ and $j$ are interchanged. In practice this exception is encountered only finitely many times, since one has irreducible polynomials depending on both $i$ and $j$. If this exception occurs only finitely many times, or if the quotient $I/J$ is bounded from above and below, then for $I > J > 1$ and $B_\lambda > 10$ the cost of using this procedure for all elements of $\mathfrak{B}_\lambda$ is

$$O\big(\frac{JB_\lambda}{\log(B_\lambda)} + \sum_{\mathfrak{p} \in \boldsymbol{B}_\lambda} \frac{IJ}{N(\mathfrak{p})}\big) = O\big(\frac{JB_\lambda}{\log(B_\lambda)} + IJ \log(\log B_\lambda)\big).$$

The first summand of the left hand side contains the initialization costs and the error made when the number of elements of $\mathbb{Z} \times \{j\} \cap \mathcal{A} \cap \mathfrak{p}$ is approximated by $I/N(\mathfrak{p})$. The transition to the right hand side is made by an application of Merten's law for number fields. For the naive method of sieving described above, this cost estimate cannot be improved. For $B_\lambda \gg I$, the first summand (containing the initialization cost per $j$) dominates. This means that it is necessary to look for a different treatment of the factor base elements with $N(\mathfrak{p}) \gg I$ if one wants to obtain a procedure with heuristic run time

$$IJ \log\log(B_\lambda). \tag{1}$$

Pollard [Pol] achieves this by calculating a reduced basis $(b_\kappa)_{\kappa=0}^1$ of the lattice $\mathfrak{p}$. Using effectively computable bounds for the coefficients $(x, y) \in \mathbb{Z}^2$ of the representation of an element $s \in \mathcal{A} \cap \mathfrak{p}$ as $s = xb_0 + yb_1$, one obtains an algorithm whose run time is, at least heuristically, given by (1). This method is also described in [BL] and [GLM] and has been used in the lattice siever, written by A. Lenstra, which was used in the RSA130–RSA155 GNFS records and in the SNFS records obtained during that time.

Our method of lattice sieving depends on the following two facts:

**Proposition 1.** *Let $\mathfrak{p} \subset \mathbb{Z}^2$ be a lattice such that $\mathbb{Z}^2/\mathfrak{p}$ is cyclic of order $p$, and such that $\mathbb{Z} \times \{0\} \cap \mathfrak{p} = p\mathbb{Z}$. Let a number $I \leq p$ be given. Then there exists a unique basis $\{(\alpha, \beta), (\gamma, \delta)\}$ of $\mathfrak{p}$ with the following properties:*

- *We have $\beta > 0$ and $\delta > 0$.*
- *We have $-I < \alpha \leq 0 \leq \gamma < I$.*
- *We have $\gamma - \alpha \geq I$.*

*If $(i, j) \in \mathfrak{p}$ satisfies $-I < i < I$ and $j \geq 0$, then $(i, j) = k(\alpha, \beta) + l(\gamma, \delta)$ with $k \geq 0$ and $l \geq 0$.*

*Proof.* Our assumptions imply the existence of an integer $r$ such that $0 \leq r < p$ and $\mathfrak{p} = \big\{(i, j) \in \mathbb{Z}^2 \;\big|\; i \equiv rj \bmod p\big\}$. We define a sequence of integer pairs by $(i_0, j_0) = (-p, 0)$, $(i_1, j_1) = (r, 1)$, and $(i_{k+1}, j_{k+1}) = (i_{k-1}, j_{k-1}) + a_k(i_k, j_k)$, where the positive integer $a_k$ is obtained by rounding $|i_{k-1}/i_k|$ towards 0. The sequence

stops if $i_k = \pm 1$. By induction, $(i_{k-1}, j_{k-1})$ and $(i_k, j_k)$ form a basis of $\mathfrak{p}$. It is easy to see that $j_k$ is positive for $k > 0$, and that the sequence $(-1)^{k+1} i_k$ is a strictly decreasing sequence of non-negative integers. Therefore, and because of $i_0 \leq -I$, there exists an integer $k > 0$ with $|i_k| < I$ and $|i_{k-1}| \geq I$. Let $a$ be the smallest integer with $|i_{k-1}| - a\,|i_k| < I$. If $k$ is odd, then $(\alpha, \beta) = (i_{k-1}, j_{k-1}) + a(i_k, j_k)$ (we have $\beta > 0$ since $a > 0$ and $j_k > 0$) and $(\gamma, \delta) = (i_k, j_k)$ are easily seen to satisfy all three conditions. For even $k$, the same holds for $(\alpha, \beta) = (i_k, j_k)$ and $(\gamma, \delta) = (i_{k-1}, j_{k-1}) + a(i_k, j_k)$. This proves the existence of a basis of $\mathfrak{p}$ with the properties described above.

Let $\{(\alpha, \beta); (\gamma, \delta)\}$ be a basis of $\mathfrak{p}$ satisfying these conditions. Also, let $(i, j) \neq 0$ satisfy the conditions of the last assertion of the proposition. Since $(i, j) \in \mathfrak{p}$, we have $(i, j) = k(\alpha, \beta) + l(\gamma, \delta)$ with integers $k$ and $l$. We have to show that both $k$ and $l$ are non-negative. If this fails to hold, then either both of them are non-positive, with one of them being negative, or they are both different from zero and have different signs. The first possibility yields the contradiction $j < 0$ since $\beta$ and $\delta$ are both positive. The second possibility leads to the contradiction

$$|k\alpha + l\gamma| = |l|\,\gamma - |k|\,\alpha \geq \gamma - \alpha \geq I,$$

where the equality holds because $k\alpha$ and $l\gamma$ have the same sign.

We have just shown that the second assertion of the proposition holds for each lattice base satisfying the assumptions of the first assertion. This implies that $(\alpha, \beta)$ coincides with the element $(i, j)$ of $[1 - I, 0] \times (0, \infty) \cap \mathfrak{p}$ for which $j$ is smallest, whereas the other basis vector is the element of $[0, I - 1] \times (0, \infty) \cap \mathfrak{p}$ for which $j$ is smallest. This proves the uniqueness assertion. □

**Proposition 2.** *Let $\mathfrak{p}$, $I$, $(\alpha, \beta)$ and $(\gamma, \delta)$ be the same as in the previous proposition. Let $A \in \mathbb{N}$ and $\tilde{\mathcal{A}} = \mathbb{Z}^2 \cap \big([A, A + I - 1] \times \mathbb{Z}\big)$. Let $(i, j)$ in $\tilde{\mathcal{A}} \cap \mathfrak{p}$. If $(i', j') \in \tilde{\mathcal{A}} \cap \mathfrak{p}$ where $j' > j$ and $j'$ is as small as possible, then*

$$(i', j') = (i, j) + \begin{cases} (\alpha, \beta) & i \geq A - \alpha \\ (\gamma, \delta) & i < A + I - \gamma \\ (\alpha, \beta) + (\gamma, \delta) & A + I - \gamma \leq i < A - \alpha \end{cases} \tag{2}$$

*Proof.* It is easy to see that precisely one of the three cases occurs, and that the right hand side of (2) is always an element of $\tilde{\mathcal{A}} \cap \mathfrak{p}$ whose second coordinate $\hat{j}$ is $> j$. If $(\tilde{i}, \tilde{j}) \in \tilde{\mathcal{A}} \cap \mathfrak{p}$ with $\tilde{j} > j$, then 1 implies

$$(\tilde{i}, \tilde{j}) = (i, j) + k(\alpha, \beta) + l(\gamma, \delta)$$

with $k \geq 0$, $l \geq 0$ and $k + l > 0$. In the first and third case, the condition $\tilde{i} < A + I$ is violated if $k = 0$ and $l > 0$. Therefore, $k > 0$ in the first and third case. Similarly, the condition $\tilde{i} \geq A$ forces $l > 0$ in the second and third case. In all three cases, we obtain $\tilde{j} \geq \hat{j}$. □

*Remark* 1. We outline some implementation details for the application of proposition 2 to lattice sieving. First of all, it is possible to restrict to the case $J \leq I/2$ by exchanging the two coordinates of $\mathcal{A}$ and using the fact that $(i, j)$ and $-(i, j)$ are equivalent number field sieve reports. Recall that $\mathcal{A} = [-I/2, I/2 - 1] \times [1, J]$. If $\mathfrak{p}$ is a factor base element with $p = N(\mathfrak{p}) \geq I$ which violates the assumption of proposition 1, then $\mathfrak{p}$ is defined by the congruence $p|j$, which contradicts $(i, j) \in \mathcal{A}$. Therefore, propositions 1 and 2 can be applied to all factor base elements with

$\mathfrak{p} \cap \mathcal{A} \neq \emptyset$ which are not treated by the line sieving procedure. For such factor base elements, one does not store the pairs $(\alpha, \beta)$ and $(\gamma, \delta)$ but the numbers $a = \beta I + \alpha$ and $c = \delta I + \gamma$. Before starting the inner sieving loop which will produce all elements of $\mathcal{A} \cap \mathfrak{p}$, one calculates the bounds $b_0 = -\alpha$ and $b_1 = I - \gamma$ from $a$ and $c$. The elements $(i, j)$ of $\mathcal{A} \cap \mathfrak{p}$ are parametrized by $x = jI + i + I/2$, and the sieve array $S$ is treated as a one-dimensional array. After calculating (by application of 2 to the pair $(0, 0)$) the smallest $x$ or reading some previously used $x$ from memory, the inner sieving loop proceeds in the following steps:

- $i \leftarrow x \% I$
- $S(x) \leftarrow S(x) + \log(N(\mathfrak{p}))$
- If $i \geq b_0$, $x \leftarrow x + a$.
- If $i < b_1$, $x \leftarrow x + c$

which are executed until $x \geq I(J + 1)$. If $I$ is a power of 2, the first step is just a bitwise `AND`. In our SNFS and GNFS records, as well as in the RSA140 and RSA155 factorizations, $I$ was a power of 2.

The first two steps are always independent of each other, and at least some substeps of steps 3 and 4 are independent of each other and of step 2. Also, steps 3 and 4 are easily implemented in a jump-free way on a modern CPU. For these reasons, and because a truncated Euclidean algorithm is probably faster than a lattice reduction in most cases, the procedure is probably very fast. However, we have no comparisons with implementations of other lattice sieving methods optimized for the same target machines.

In reality, the sieving region is broken into blocks of size $L_1$ equal to the L1 cache, and for primes exceeding the cache size the second step is not carried out directly. Instead, the remainder of $x$ modulo $L_1$ is pushed to the appropriate entry of a one-dimensional array of stacks containing one stack for each L1 block. When the $i$-th block of size $L_1$ is sieved, the entries $x_{i,j}$ of the $i$-th stack are read and the appropriate value is added at offset $x_{i,j}$ of an array of bytes of size $L_1$. To reduce the amount of memory used by this method, and to make it more cache efficient, this procedure is actually broken into several steps corresponding to several intermediate bounds on the factor base primes. For instance, for factor base primes $L_1 \leq p \leq 16L_1$, the values $x_{i,j}$ are only stored for 16 adjacent values of $i$. For larger primes, more adjacent values of $i$ have to be considered in order to retain the log log-behaviour of Merten's law.

To facilitate the trial division sieve ([GLM]), we also store a hint which allows the reconstruction of the factor base element. The trial division sieve is carried out seperately for each of the subregions of size $L_1$ of the sieve region.

*Remark* 2. Let $(i_k, j_k)$ be the sequence considered in the proof of proposition 1. It is easy to see that the quotients $n_k/j_k$, where $n_k = (j_k r - i_k)/p$, are the continued fraction approximations to $r/p$. The pair $(\alpha, \beta)$ or $(\gamma, \delta)$ whose second coordinate is smallest corresponds to the first diophantine approximation $n/m = [0, a_1, \ldots, a_{k-1}]$ to $r/p$ with $\left|\frac{mr}{p} - n\right| < mI$. The other pair is only a one-sided diophantine approximation to $r/p$, and corresponds to $[0, a_1, \ldots, a_{k-1}, a]$ with $1 \leq a \leq a_k$. This explains the title of the paper and explains why proposition 1 is a modification of the well-known theorem about diophantine approximations and continued fractions.

## 2. Splitting the sieving region

A simple parallelization of lattice sieving may split the sieving region $\mathcal{A} = \mathbb{Z}^2 \cap ([-I/2, I/2-1] \times [1, J])$ into pieces $\mathcal{A}_a = \mathbb{Z}^2 \cap ([-I/2, I/2-1] \times [aJ_1+1, (a+1)J_1])$ with $0 \le a < K = J/J_1$, where $J_1$ divides $J$. Each of the $K$ nodes stores all factor base elements with $N(\mathfrak{p}) < IJ_1$. The other factor base elements are distributed among the $K$ nodes. The $a$-th node calculates all elements of $\mathcal{A}_a \cap \mathfrak{p}$ for the factor base elements with $N(\mathfrak{p}) < IJ_1$. For the other factor base elements, it receives part of the information from the other nodes. For its own share of the factor base elements with $N(\mathfrak{p}) \ge IJ_1$, the node calculates the elements $z$ of $\mathfrak{p} \cap \mathcal{A}$ and sends them to the destination node with $z \in \mathcal{A}_a$. For the factor base elements with $N(\mathfrak{p}) < IJ_1$, the $a$-th node has to calculate the element of $\mathcal{A}_a \cap \mathfrak{p}$ whose second coordinate is smallest. This is easily seen to be a special case of the following problem.

Let $\mathfrak{p}$, $\mathfrak{a} = (\alpha, \beta)$, $\mathfrak{c} = (\gamma, \delta)$ and $\tilde{\mathcal{A}}$ be the same as in proposition 2. Let $\mathfrak{i} = (i, j) \in \tilde{\mathcal{A}}$ and $D > 0$ be given, and let $\tilde{\mathfrak{i}} = (\tilde{i}, \tilde{j})$ be the element of $\tilde{\mathcal{A}} \cap \mathfrak{p}$ whose second coordinate is the smallest possible with $\tilde{j} > j + D$. To determine $\tilde{\mathfrak{i}}$, one may proceed in the following steps:

**Step 1:** Let $(0, D) = \xi\mathfrak{a} + \upsilon\mathfrak{c}$, and let $x$ and $y$ be obtained by rounding the rational numbers $\xi$ and $\upsilon$ (which are easily seen to be positive) towards zero. Let $\mathfrak{i}_1 = (i_1, j_1) = \mathfrak{i} + (x+1)\mathfrak{a} + y\mathfrak{c}$, $\mathfrak{i}_2 = (i_2, j_2) = \mathfrak{i} + x\mathfrak{a} + (y+1)\mathfrak{c}$, and $\mathfrak{i}_3 = (i_3, j_3) = \mathfrak{i} + (x+1)\mathfrak{a} + (y+1)\mathfrak{c}$. We have $i_1 \le i < A + I$ and $i_2 \ge i \ge A$. If $i_1 \ge A$, then output $\mathfrak{i}_1$ if $j_1 > j + D$ and go to Step 2 if $j_1 \le j + D$. If $i_2 < A + I$, go to Step 3 if $j_2 \le j + D$ and output $\mathfrak{i}_2$ if $j_2 > j + D$. If $i_1 < A$ and $i_2 \ge A + I$, output $\mathfrak{i}_3$.

**Step 2:** We have $\mathfrak{i}_1 \in \tilde{\mathcal{A}} \cap \mathfrak{p}$. An iterated application of proposition 2 to this pair shows that $(\tilde{i}, \tilde{j})$ can be found as follows: Let $l$ be the largest non-negative integer with $i_1 + l\alpha \ge A$ and $j_1 + l\beta \le j + D$. If $i_1 + (l+1)\alpha \ge A$, output $\mathfrak{i}_1 + (l+1)\mathfrak{a}$. Otherwise, output $\mathfrak{i}_3 + l\mathfrak{a}$ if $i_3 + l\alpha < A + I$ or $\mathfrak{i}_3 + (l+1)\mathfrak{a}$ if $i_3 + l\alpha \ge A + I$.

**Step 3:** We apply a similar procedure to $\mathfrak{i}_2$: Let $l \ge 0$ be the largest integer with $i_2 + l\gamma < A + I$ and $j_2 + l\delta \le j + D$. Output $\mathfrak{i}_2 + (l+1)\mathfrak{c}$ if the first entry of this pair is $< A + I$. Otherwise, output $\mathfrak{i}_3 + l\mathfrak{c}$ or $\mathfrak{i}_3 + (l+1)\mathfrak{c}$, preferring the first pair if its first entry is $\ge A$.

In steps 2 and 3, we repeatedly apply proposition 2 to a pair $\mathfrak{i} + x'\mathfrak{a} + y'\mathfrak{c}$ where the coefficient of one of the basis vectors satisfies $x' \ge \xi$ or $y' \ge \upsilon$. As soon as the other basis vector is added, the repeated application of (2) will produce an element of $\tilde{\mathcal{A}} \cap \mathfrak{p}$ with second coordinate $> j + D$, but it may do so earlier. This justifies steps 2 and 3, and it remains to justify step 1 in the cases where it produces the output.

If $i_1 \ge A$, then $\mathfrak{i}_1 \in \tilde{\mathcal{A}} \cap \mathfrak{p}$, and every element $\hat{\mathfrak{i}} = (\hat{i}, \hat{j})$ of $\tilde{\mathcal{A}} \cap \mathfrak{p}$ with $\hat{j} < j_1$ has, by application of proposition 1 to $\mathfrak{i}_1 - \hat{\mathfrak{i}}$, the form $\mathfrak{i} + \hat{x}\mathfrak{a} + \hat{y}\mathfrak{c}$ with $\hat{x} \le \xi$ and $\hat{y} \le \upsilon$. This demonstrates $\hat{j} \le j + D$, and implies $\tilde{\mathfrak{i}} = \mathfrak{i}_1$ if $\mathfrak{i}_1 \in \tilde{\mathcal{A}} \cap \mathfrak{p}$ and $j_1 > j + D$. A similar consideration justifies step 1 if $i_2 < A + I$.

It remains to justify step 1 if $\mathfrak{i}_1 \notin \tilde{\mathcal{A}}$ and $\mathfrak{i}_2 \notin \tilde{\mathcal{A}}$. In this case, $i_3 = i_1 + \gamma < A + \gamma < A + I$, $i_3 = i_2 + \alpha \ge A + I + \alpha > A$, and $j_3 > j + \xi\beta + \upsilon\delta = j + D$, which proves $\mathfrak{i}_3 \in \tilde{\mathcal{A}} \cap \mathfrak{p}$ with $j_3 > j + D$. If $\hat{\mathfrak{i}} \in \tilde{\mathcal{A}} \cap \mathfrak{p}$ with second coordinate $\hat{j} < j_3$, then an application of proposition 1 to $\mathfrak{i}_3 - \hat{\mathfrak{i}}$ gives $\hat{\mathfrak{i}} = \mathfrak{i} + \hat{x}\mathfrak{a} + \hat{y}\mathfrak{c}$ with $\hat{x} \le x + 1$ and

$\hat{y} \le y+1$, where one of these inequalities must be strict. If only the first inequality is strict, we obtain the contradiction $\hat{i} \ge i_2 \ge A+I$. If the second inequality is strict while the first one is not, we have the contradiction $\hat{i} \le i_1 < A$. Since both inequalities are strict, we have $\hat{x} \le \xi$ and $\hat{y} \le \upsilon$, which implies $\hat{j} \le j+D$. The justification of the above procedure is complete.

## 3. Parallelization of the lattice siever

If just a few nodes are involved, then the all-to-all communication scheme explained in the last subsection may be appropriate. This may change for larger projects like RSA1024, which may need factor base sizes up to $10^{10}$. This means that up to a few hundred nodes are involved. It is the aim of this section to analyse the Bernstein cost function for a large multicomputer dedicated to sieving.

For line sieving, we suppose that the "mathematical" sieving region is $\mathfrak{a}+([0,A]\cap\mathbb{Z})\times\mathbb{Z}$, where $A=2^{k+l}$ is a power of two and $\mathfrak{a}$ is a vector in the domain of definition of the polynomial pair. It is mapped bijectively to the interval $\tilde{\mathcal{A}}=\mathbb{Z}\cap[0,2^{k+l}-1]$ which is divided into $2^l$ pieces of size $2^k$. In the case of the lattice siever, the same is achieved by mapping $\mathcal{A}=[-I/2,I/2-1]\times[1,J]\cap\mathbb{Z}^2$ bijectively to $\tilde{\mathcal{A}}$, mapping $(i,j)$ to $(j-1)*I+i+I/2$. In this case, it is assumed that $2^{k+l}=IJ$. For the sake of simplicity, we always assume $N(\mathfrak{p})<2^{k+l}$ for factor base elements. This assumption is reasonable up to a factor of at most $\log(k+l)$, since otherwise initialization costs start to dominate over sieving costs. For a factor base element $\mathfrak{p}$, $\tilde{\mathcal{A}}_\mathfrak{p}$ denotes the image of $\mathcal{A}\cap\mathfrak{p}$ under the bijection $\mathcal{A}\xrightarrow{\cong}\tilde{\mathcal{A}}$.

The parallel computer consists of $l+1$ layers, each containing $2^l$ nodes. The $b$-th layer, $0\le b\le l$, contains the nodes $\mathfrak{N}_{a,b}$, $0\le a<2^l$. If $b>0$, then $\mathfrak{N}_{a,b}$ has an outgoing connection to $\mathfrak{N}_{a,b-1}$ and to $\mathfrak{N}_{a\pm2^{b-1},b-1}$, where in $a\pm2^{b-1}$ the $+$-sign is chosen if the remainder of the division $a/2^b$ is $<2^{b-1}$, and the $-$-sign is chosen if this remainder is $\ge 2^{b-1}$. Each node only looks at a part $\mathfrak{B}_\lambda^{(a,b)}$ of the factor base and a part $\tilde{\mathcal{A}}_{a,b}$ of the sieving interval. We put $\tilde{\mathcal{A}}_{a,0}=\mathbb{Z}\cap[a2^k,(a+1)2^k-1]$. For $b>0$, we put $\tilde{\mathcal{A}}_{a,b}=\tilde{\mathcal{A}}_{a,b-1}\cup\tilde{\mathcal{A}}_{a',b-1}$ where $\mathfrak{N}_{a',b-1}$, $a'\ne a$ is the unique node in layer $b-1$ connected to $\mathfrak{N}_{a,b}$. Let $\mathfrak{B}_\lambda^{(0)}$ consist of all $\mathfrak{p}\in\mathfrak{B}_\lambda$ such that $N(\mathfrak{p})<2^k$. For $b>0$, let $\mathfrak{B}_\lambda^{(b)}$ consist of all $\mathfrak{p}\in\mathcal{B}_\lambda$ such that $2^{k+b-1}\le N(\mathfrak{p})<2^{k+b}$. We assume that, for each $\mathfrak{p}\in\mathfrak{B}_\lambda^{(b)}$, a random number $\mathfrak{r}_\mathfrak{p}\in\mathbb{Z}\cap[0,2^b-1]$ is calculated such that the $\mathfrak{r}_\mathfrak{p}$ are equidistributed and statistically independent. This means that for $j$ distinct elements of $\mathfrak{B}_\lambda^{(b)}$ and $(i_1,\dots,i_j)\in[0,2^b-1]$ the probability of $\mathfrak{r}_{\mathfrak{p}_m}=i_m$ for each $m$ with $1\le m\le j$ equals $2^{-bj}$. We define $\mathfrak{B}_\lambda^{(a,b)}$ to be the set of all $\mathfrak{p}\in\mathfrak{B}_\lambda^{(b)}$ for which $\mathfrak{r}_\mathfrak{p}$ is equal to the remainder of the division of $a$ by $2^b$. It is easy to see that for each pair $(x,\mathfrak{p})\in\tilde{\mathcal{A}}\times\mathfrak{B}_\lambda$, there is precisely one node $\mathfrak{N}_{a,b}$ with $x\in\tilde{\mathcal{A}}_{a,b}$ and $\mathfrak{p}\in\mathfrak{B}_\lambda^{(a,b)}$. and $\tilde{\mathcal{A}}_{a,b,\mathfrak{p}}=\tilde{\mathcal{A}}_\mathfrak{p}\cap\tilde{\mathcal{A}}_{a,b}$ for all factor base elements and $(a,b)$-pairs. The preimage of $\tilde{\mathcal{A}}_{a,b}$ in $\mathcal{A}$ will be denoted by $\mathcal{A}_{a,b}$.

The hosts $\mathfrak{N}_{a,0}$ of the bottom layer actually allocate memory for an array of bytes $S[x]$ with $x\in\tilde{\mathcal{A}}_{a,0}$. All hosts $\mathfrak{N}_{a,b}$ allocate memory for the elements of $\mathfrak{B}_\lambda^{(a,b)}$. In addition, all hosts have to allocate some buffers for the communication described below.

The sieving of a special $q$ by the parallel computer takes $l+2$ steps which we describe now, at least in the case of line sieving. In step 0, $\mathcal{N}_{a,b}$ calculates the set $\mathfrak{S}_{a,b}$ of all pairs $(x,\mathfrak{p})$ with $\mathfrak{p}\in\mathfrak{B}_\lambda^{(a,b)}$ and $x\in\mathcal{A}_{a,b,\mathfrak{p}}$. If $b=0$, then $\log(N(\mathfrak{p}))$ is

added to $S[x]$ for each pair $(x, \mathfrak{p}) \in \mathfrak{S}_{a,b}$. Otherwise, $\mathfrak{N}_{a,b}$ sends $(x, \mathfrak{p})$ to $\mathfrak{N}_{a',b-1}$, where $a'$ is determined by the conditions that the target node is one of the two nodes to which $\mathfrak{N}_{a,b}$ has an outgoing connection, and that $x \in \tilde{\mathcal{A}}_{a',b-1}$.

In step $s$, where $1 \leq s \leq l$, the nodes $\mathfrak{N}_{a,b}$ with $b + s \leq l$ have received from one of their two links in layer $b + 1$ all pairs $(x, \mathfrak{p}) \in \mathfrak{S}_{a,b+s,b}$, the subset of $\bigcup_{a' \equiv a \bmod 2^b} \mathfrak{S}_{a',b+s}$ defined by the condition $x \in \tilde{\mathcal{A}}_{a,b}$. It deals with these pairs in the same way as with the pairs which it calculated in step 0. If $b = 0$, $\log(N(\mathfrak{p}))$ is added to $S[x]$. Otherwise the pair $(x, \mathfrak{p})$ is sent to the node $\mathfrak{N}_{a',b-1}$ to which $\mathfrak{N}_{a,b}$ has an outgoing connection and for which $x \in \tilde{\mathcal{A}}_{a',b-1}$. The nodes with $b > l - s$ are idle in this step.

In step $l + 1$, the nodes of the bottom layer determine the set of all $x \in \tilde{\mathcal{A}}_{a,0}$ for which $S[x]$ exceeds a certain threshold. For such $x$, they calculate the value of the norm polynomials, test them for smoothness, and output the sieve report if the norm polynomial is found to be smooth. Countless variations to this scheme may be considered. First of all, one has to decide whether the above procedure is carried out for $\lambda = 0$ only or for both values of $\lambda$. If sieving is only done for $\lambda = 0$ (as proposed by Coppersmith [Cop] and Bernstein [Ber]), then all survivors of sieving on this side have to be fed to the elliptic curve test. Otherwise, the nodes of the bottom layer have to allocate two (or more, if Coppersmith's multi-polynomial version is used) sieve intervals. In this case, the inter-node communication also has to keep track of $\lambda$. It is also possible to use a trial division sieve if, for some cut-off parameter $b'$, the nodes of the bottom layer store the elements of $\bigcup_{b=b'}^{l} \mathfrak{S}_{a,b,0}$ instead of just adding $\log(N(\mathfrak{p}))$ to $S[x]$.

Recall our announcement that the above procedure needs some modification if the lattice siever is to be parallelized. Before we describe this modification, we give an estimate of Bernstein's cost function for the parallelized line siever. We fix $k$ and assume $l \to \infty$. Let $\mathfrak{n}$ be the polynomial norm and $\mathfrak{d}$ be the polynomial degree. By Chebychev's prime number theorem, the number of elements in $\mathfrak{B}_\lambda^{(b)}$ is $\ll \frac{\mathfrak{d} 2^{k+b}}{k+b}$ unconditionally, and $\ll \frac{2^{k+b}}{k+b}$ heuristically.[1] In any case, their number is $\ll 2^{k+b}$ up to factors which are at most polynomial in $\mathfrak{d}$. The number of elements $\tilde{\mathcal{A}}_{a,b,\mathfrak{p}}$ with $\mathfrak{p} \in \mathfrak{B}_\lambda^{(b)}$ is $O(1)$ with implied constant depending on $k$ for $b = 0$ and independent of $k$ if $b > 0$ . If the elements of $\mathfrak{B}_\lambda^{(b)}$ are evenly distributed among the nodes of layer $b$ which treat the same subset of $\tilde{\mathcal{A}}$, then the amount of memory of $\mathfrak{N}_{a,b}$ needed for storing $\mathfrak{B}_\lambda^{(a,b)}$ and $\mathfrak{S}_{a,b}$, as well as the execution time of step 0, are of at most polynomial growth in $\mathfrak{d}$ and $l$. The fact that we used a random number generator to distribute $\mathfrak{B}_\lambda^{(b)}$ over the nodes introduces some difficulties which are easily dealt with since they are similar to, but easier than the problems encountered in the later steps.

Since it is not easy to prove useful unconditional inequalities for the number of elements in $\mathfrak{S}_{a,b',b}$ with $b' > b$, the cost of steps 1 to $l$ is more difficult to estimate. It is only possible to prove a probabilistic result. Let

$$\mathfrak{S}^{\mathrm{tot}}_{a,b',b} = \left\{ (x, \mathfrak{p}) \;\middle|\; \mathfrak{p} \in \mathfrak{B}_\lambda^{(b')} \text{ and } x \in \tilde{\mathcal{A}}_{a,b,\mathfrak{p}} \right\}$$

[1] We write $f(x) \ll g(x)$ if there exists a constant $C$ such that $0 \leq f(x) \leq Cg(x)$ for all $x$ of the domain of definition of both functions.

This set does not change if $a$ varies in a certain block of $2^b$ consecutive integers, and is the union of $\mathfrak{S}_{a',b',b}$ with $a'$ varying in this block:

$$\mathfrak{S}^{\text{tot}}_{a,b',b} = \bigcup_{a'=a_0 2^b}^{a_0 2^{b+1}-1} \mathfrak{S}_{a',b',b} \qquad \text{for } 2^b a_0 \le a < 2^{b+1} a_0.$$

To estimate the number of elements in this set, note that its elements are related to prime divisors of the polynomial values at the $2^{b+k}$ elements of $\mathcal{A}_{a,b}$. Since each prime factor is $\ge 2^{b'+k}$, the number of elements in $\mathfrak{S}^{\text{tot}}_{a,b',b}$ is

$$N_{a,b',b} := \#\big(\mathfrak{S}^{\text{tot}}_{a,b',b}\big) \ll \frac{(\mathfrak{d}(l + \log|\mathfrak{a}|) + \log\mathfrak{n})2^{k+b}}{k+b'}. \tag{3}$$

Note that the sets $\mathfrak{S}^{\text{tot}}_{a,b',b}$ depend only on the sieving task, not on the random distribution of the factor base elements over the nodes in layer $b'$. This is no longer the case for $\mathfrak{S}_{a,b',b}$: An element $(x,\mathfrak{p})$ of $\mathfrak{S}^{\text{tot}}_{a,b',b}$ belongs to $\mathfrak{S}_{a,b',b}$ if and only if $2^b$ divides $r_\mathfrak{p} - a$, where $r_\mathfrak{p}$ is the random variable introduced in the description of $\mathfrak{B}^{(a',b')}_\lambda$. This means that the random variables $r_{(x,\mathfrak{p})}$ associating to $(x,\mathfrak{p}) \in \mathfrak{S}^{\text{tot}}_{a',b',b}$ the number $a$ with $x \in \mathfrak{S}_{a,b',b}$ are equidistributed over a block of $2^b$ consecutive integers. Since we are working with the line siever and since the elements of $\mathfrak{B}^{(b')}_\lambda$ satisfy $N(\mathfrak{p}) > 2^{k+b}$, it is easy to see for each $\mathfrak{p} \in \mathfrak{B}^{(b')}_\lambda$ there is at most one $(x,\mathfrak{p}) \in \mathfrak{S}^{\text{tot}}_{a,b',b}$. Since the $r_\mathfrak{p}$ are statistically independent, this implies that the $r_{(x,\mathfrak{p})}$ are also statistically independent.

Now we assume that the communication buffer of node $\mathfrak{N}_{a,b}$, which is allocated before sieving starts, contains at least

$$M_{a,b',b} = \left\lceil \frac{l((l + \log(|\mathfrak{a}|))\mathfrak{d} + \log\mathfrak{n})2^k}{k+b'} \right\rceil \tag{4}$$

elements. It is assumed that the nodes are programmed in such a way that they never write past the end of their communication buffers. Instead, they abort a sieving for which the communication buffer overflows, or split the transmission into several passes if this occurs. For the sake of simplicity, we assume the first possibility. The probability that a sieving task is aborted because of buffer overflow in node $\mathfrak{N}_{a,b}$ in step $b' - b$ is

$$\begin{aligned} p_{\text{abort}}(a,b',b) &\le \sum_{s=M_{a,b',b}+1}^{N_{a,b',b}} 2^{-bs}(1-2^{-b})^{N_{a,b',b}-s}\binom{N_{a,b',b}}{s} \\ &\ll \sum_{s=M_{a,b',b}+1}^{N_{a,b',b}} 2^{-bs}\frac{N_{a,b',b}^{N_{a,b',b}}}{s^s(N_{a,b',b}-s)^{N_{a,b',b}-s}} \\ &\ll \sum_{s=M_{a,b',b}+1}^{N_{a,b',b}} \Big(2^{-bs}\frac{e\cdot N_{a,b',b}}{s}\Big)^s \qquad (5) \\ &\le \Big(\frac{C}{l}\Big)^l \end{aligned}$$

for some constant $C$. The total probability of a sieving task being aborted is bounded by $(l+1)^2 2^l$ times this expression, and is easily seen to tend to zero as $l \to \infty$.

The cost of building a node which is able to allocate a communication buffer of at least (4) elements of size $2 * (k + l)$ bits grows at most polynomially in $l$, $\log(|\mathfrak{a}|\,\mathfrak{n})$ and $\mathfrak{d}$. If the communication buffer does not overflow, then the execution time for each of the steps is also bounded by a multiple of the maximum of (4) over $0 \le b < b' \le l$ times some power of $k + l$. We arrive at

**Proposition 3.** *Using the parallel siever described above, the Bernstein cost function for executing a line sieving task of size $2^{k+l}$, with factor base bound $< 2^{k+l}$, polynomial norm $\mathfrak{n}$ and degree $\mathfrak{d}$ and with probability of success tending to 1 as $l \to \infty$ uniformly in $\mathfrak{n}$ and $\mathfrak{d}$, is*

$$\ll_k \big(l + \mathfrak{d} + \log(|\mathfrak{a}|\,\mathfrak{n})\big)^{O(1)} 2^l.$$

The parallelization of the lattice siever is similar to the case of the line siever, with one exception. For the line siever, we made use of the fact that the number of elements of $\tilde{\mathcal{A}}_{a,b,\mathfrak{p}}$ is $\le 1$ if $N(\mathfrak{p}) \ge 2^{k+b}$. This still holds for the lattice siever if $2^{k+b} \le I$. Otherwise, the number of "hits" of this factor base element in $\tilde{\mathcal{A}}_{a,b}$ may be as large as $2^{k+b}/I$, even if $N(\mathfrak{p})$ is much larger than $2^{k+b}$. This may occur if one of the two basis vectors in proposition 1 is extremely short. For this reason, it is necessary to fix some constant $\mathfrak{K}$ and modify the procedure as follows:

- In step 0, the node $\mathfrak{N}_{a,b}$ applies the procedure explained in the proof of proposition 1 to the elements of $\mathfrak{B}_\lambda^{(a,b)}$. It determines the smallest element of $\tilde{\mathcal{A}}_{a,b,\mathfrak{p}}$, using the procedure outlined in the second subsection. If $b = 0$, it carries out the sieving with this factor base element. If $b > 0$, there are two integers $a'$ for which $\mathfrak{N}_{a,b}$ has an outgoing connection to $\mathfrak{N}_{a',b-1}$. For each of them, $\mathfrak{N}_{a,b}$ determines whether the number of elements of $\tilde{\mathcal{A}}_{a',b-1,\mathfrak{p}}$ is $< \mathfrak{K}$. In this case, it sends the pairs $(x, \mathfrak{p})$ with $x \in \tilde{\mathcal{A}}_{a',b-1,\mathfrak{p}}$ to $\mathfrak{N}_{a',b-1}$. Otherwise, it treats $\mathfrak{p}$ as degenerate and sends only $\mathfrak{p}$ itself. This exception can be encoded, for instance, by sending the pair $(x, \mathfrak{p})$ with some $x$ outside $\mathcal{A}_{a',b-1}$.
- In steps 1 through $l$, the hosts deal with the $(x, \mathfrak{p})$ pairs they receive in precisely the same way as for line sieving. If a host receives an exceptional factor base element $\mathfrak{p}$, it deals with $\mathfrak{p}$ in the same way as with the elements of $\mathfrak{B}_\lambda^{(a,b)}$ in step 0.

The last step works in the same way as for line sieving. The constant $\mathfrak{K}$ could be set equal to 1 (such that all factor base elements are treated as exceptional) without seriously changing the outcome of our asymptotic analysis, but in practice it is larger and its optimal value may increase with $l$. In any case, the procedure makes sure that no node, with the possible exception of the nodes in layer 0, spends more than some power of $(k + l)$ of computing time or memory on a single factor base element which it treats. Defining $\mathfrak{S}^{\mathrm{tot}}_{a,b',b}$ and $\mathfrak{S}_{a,b',b}$ in the same way as for line sieving, our considerations for (3) imply

$$\#\big(\mathfrak{S}^{\mathrm{tot}}_{a,b',b}\big) \ll \frac{(\mathfrak{d}(l + \log q) + \log \mathfrak{n})2^{k+b}}{k + b'} \tag{6}$$

for the lattice siever. In particular, the number of factor base elements contributing to $\mathfrak{S}^{\mathrm{tot}}_{a,b',b}$ is bounded by the right hand of this expression. If at least

$$M_{a,b',b} = \left\lceil (\mathfrak{K} + 1)\frac{l((l + \log q)\mathfrak{d} + \log \mathfrak{n})2^k}{k + b'} \right\rceil \tag{7}$$

elements are allocated in the communication buffer, then our analysis for the line siever goes through. For the nodes of layer 0, we have $\mathfrak{S}_{a,0}^{\mathrm{tot}} = \mathfrak{S}_{a,0}$, and a direct application of (6) proves that the computing cost for the nodes of layer 0 stays acceptable, although some factor base elements with $\mathfrak{N}(\mathfrak{q}) > I$ may, in the case where $I < 2^k$, give an exceptionally high contribution to it. We obtain:

**Proposition 4.** *Using the parallel siever described above, the Bernstein cost function for executing a lattice sieving task of size $2^{k+l}$, with factor base bound $< 2^{k+l}$, polynomial norm $\mathfrak{n}$ and degree $\mathfrak{d}$ and with probability of success tending to 1 as $l \to \infty$ uniformly in $\mathfrak{n}$ and $\mathfrak{d}$, is*

$$\ll_k \big(l + \mathfrak{d} + \log(q\mathfrak{n})\big)^{O(1)} 2^l,$$

*where $q$ is the special $q$ treated by the siever.*

*Remark* 3. In a GNFS factorization, $k + l$, $\mathfrak{d}$ and $\log \mathfrak{n}$, $\log(q)$ and $\log(|\mathfrak{a}|)$ are all proportional to some power of the logarithm of the number which is to be factored. This gives a cost estimate

$$\ll (k + l)^{O(1)} 2^{k+l}.$$

Using elliptic curve smoothness tests, Bernstein also achieves cost

$$2^{(k+l)(1+r_{k+l})}$$

with $r_n = o(1)$ as $n \to \infty$, but in his case $nr_n$ grows like a multiple of $\sqrt{n}$ since the elliptic curve method is used for smoothness tests. This means that his sieving method is asymptotically inferior to the straightforward parallelization presented here. Of course, at some point physical problems will prevent the realization of our design because the geometry of three-dimensional Euclidean space does not permit the building of arbitrarily large butterfly multicomputers with uniformly bounded communication costs per direct link and unit of information. For projects of a huge size, the communication lines of the butterfly multicomputer will become so long that, Bernstein's design will win but we expect that this does not occur in the range which is currently feasible, even for the worlds most powerful states. For instance, the multicomputers needed to complete a sieving task for RSA1024 within a few minutes would probably only be medium sized by modern standards, and it should not be difficult to realize the design using standard fast network equipment. Larger projects will require larger clusters, but as long as the latency time of the communication is at most a few seconds and the throughput rate remains as decent as with contemporary LINUX clusters, neither the cost per node nor the time required to complete a single sieving task (as opposed to the total number of sieving tasks) will explode. We do not expect ECM smoothness testing to break even unless the cryptanalyst has the resources to fill his entire country with powerful multicomputers, each of which has the size of a major city.

For smaller projects, the difference between the two methods is similar to the difference between the MPQS method and a CONFRAC implementation using elliptic curve smoothness tests.

*Remark* 4. Modifications to the above scheme are, of course, possible. It is possible to introduce more than two ramifications per node. It is also possible to "project" the entire multicomputer to its bottom line. This increases the cost per node by a factor of $l$, while it decreases the number of nodes by the same factor. The computing time per special $q$ will, however, increase.

## References

[Ber] D. J. Bernstein, *Circuits for Integer Factorization: A Proposal*, Manuscript, November 2001.

`http://cr.yp.to/papers.html#nfscircuit`

[BL] D. J. Bernstein and A.K. Lenstra, *A general number field sieve implementation*, in [LL], 103-126

[Cop] D. Coppersmith, *Modifications to the Number Field Sieve*, J. of Cryptology **6**, 1993, 169-180.

[GLM] R. A. Golliver, A. K. Lenstra and K. S. McCurley, *Lattice sieving and trial division*, in: Algorithmic Number Theory (ed. by L. M. Adleman, M.-D. Huang), LNCS **877**, Springer, 1994, 18–27.

[LL] A.K. Lenstra and H.W. Lenstra, Jr. (eds.), *The Development of the Number Field Sieve*, Lecture Notes in Math. **1554**, Springer, 1993.

[Pol] J. M. Pollard, *The lattice sieve*, in [LL], 43-49

University of Bonn, Department of Mathematics, Beringstrasse 1, D-53115 Bonn, Germany

*E-mail address*: {`franke,thor`}`@math.uni-bonn.de`