MATHEMATICS OF COMPUTATION
Volume 75, Number 256, October 2006, Pages 2037–2047
S 0025-5718(06)01870-9
Article electronically published on June 28, 2006

ON POLYNOMIAL SELECTION FOR THE GENERAL NUMBER FIELD SIEVE

THORSTEN KLEINJUNG

Abstract. The general number field sieve (GNFS) is the asymptotically fastest algorithm for factoring large integers. Its runtime depends on a good choice of a polynomial pair. In this article we present an improvement of the polynomial selection method of Montgomery and Murphy which has been used in recent GNFS records.

1. The polynomial selection method of Montgomery and Murphy

In this section we briefly discuss the problem of polynomial selection for GNFS. We also sketch the polynomial selection method of Montgomery and Murphy.

The first step in GNFS (see [3]) for factoring an integer N consists in the choice of two coprime polynomials f_1 and f_2 sharing a common root modulo N. If we denote the corresponding homogenized polynomials by F_1, resp. F_2, the next (and most time consuming) step in GNFS consists in finding many pairs $(a, b) \in \mathbb{Z}^2$ of coprime integers for which both values $F_i(a, b)$, $i = 1, 2$, are products of primes below some smoothness bounds B_i, $i = 1, 2$ (we will refer to these pairs as sieve reports). This is usually done by a sieving procedure which identifies (most of) these pairs in some region $\mathcal{A} \subset \mathbb{Z}^2$. In the case of line sieving $\mathcal{A}$ is of the form $[-A, A] \times [1, B] \cap \mathbb{Z}^2$ for some A and B. For lattice sieving the form of this region is more complicated, but we could use a rectangle as above as an approximation. The sieving region $\mathcal{A}$ and the smoothness bounds B_i, $i = 1, 2$, are chosen such that one finds approximately $\pi(B_1) + \pi(B_2)$ sieve reports ($\pi(x)$ denotes the number of primes below x). The time spent for sieving mainly depends on the size of the region $\mathcal{A}$, i.e., $2AB$. So we are left with two problems for the polynomial selection phase: how to find such polynomial pairs and, having found more than one, how to select a polynomial pair which minimizes sieving time.

Both problems are addressed in several articles ([4], [5], [6]). We give a short description of the results of these articles. Let $\rho(x)$ be Dickman's function which roughly is the probability that the largest prime factor of a natural number n is at most $n^{\frac{1}{x}}$. A first approximation for the number of sieve reports is given by

$$\frac{6}{\pi^2} \int_{\mathcal{A}} \rho\Big(\frac{\log(F_1(x, y))}{\log(B_1)}\Big) \rho\Big(\frac{\log(F_2(x, y))}{\log(B_2)}\Big) dx dy$$

Received by the editor December 22, 2004 and, in revised form, June 22, 2005.
2000 *Mathematics Subject Classification.* Primary 11Y05, 11Y16.
Key words and phrases. Integer factorization, GNFS, polynomial selection.

(the factor $\frac{6}{\pi^2}$ takes the probability of two integers being coprime into account). This approximation can be refined by considering the number of roots of F_i, $i = 1, 2$, modulo small primes. Let $r(F, p)$ be the number of linear factors of the homogeneous polynomial F modulo p and let

$$\alpha_i = \sum_{p \text{ small}} \left(1 - r(F_i, p)\frac{p}{p+1}\right)\frac{\log(p)}{p-1}.$$

Then a better approximation for the number of sieve reports is given by

$$\frac{6}{\pi^2}\int_{\mathcal{A}} \rho\Big(\frac{\log(F_1(x,y)) + \alpha_1}{\log(B_1)}\Big)\rho\Big(\frac{\log(F_2(x,y)) + \alpha_2}{\log(B_2)}\Big)dxdy.$$

Since this expression is difficult to compute, one needs simpler approximations. Note that we only need a method to rank several polynomial pairs, since we are only interested in finding the best one. We now assume that f_2 is of degree one (which implies that α_2 does not depend on f_2) and that $\log(F_2(x, y))$ does not vary much over the sieving region (this will be the case for the polynomials in the algorithm below). Then the term $\rho((\log(F_2(x, y)) + \alpha_2)/\log(B_2))$ can be omitted so that the integrand only depends on f_1. A further simplification consists in considering

$$\alpha_1 + \frac{1}{2}\log\left(\int_{\mathcal{A}} F_1^2(x, y)dxdy\right).$$

Here the contribution from the left summand α_1 is called root property and the contribution from the right summand is called size property. Note that we want to minimize this expression, whereas we want to maximize the previously given approximations.

Before outlining the algorithm of Montgomery and Murphy, we have to discuss some methods to improve the quality of a given polynomial pair. From a coprime polynomial pair (f_1, f_2) sharing a common root modulo N, we can produce other pairs by two methods:

(1) we can translate it by an integer t getting the pair $(\tilde{f}_1, \tilde{f}_2)$ where $\tilde{f}_i(x+t) = f_i(x)$, or
(2) we can add a $\mathbb{Z}[x]$-multiple of one polynomial to the other.

Translation preserves the value of α_1, whilst the second method may change it. Another method to optimize the quality is to change the shape of the sieving region $\mathcal{A}$ (without changing the area of $\mathcal{A}$). A rectangle of given area depends only on the ratio $s = \frac{A}{B}$ which we will call the skewness of the sieving region. Changing the skewness also preserves α_1.

We now sketch the algorithm of Montgomery and Murphy. Let the number N, a degree d, and a bound $a_{d,\max}$ for the leading coefficient be given, and set $a_d = 0$. Then execute the following steps:

- Choose the next good a_d (good means that a_d has some small prime divisors). If $a_d > a_{d,\max}$ terminate the algorithm.
- Set $m = \left\lfloor \sqrt[d]{\frac{N}{a_d}} \right\rfloor$ and determine the next two coefficients a_{d-1}, a_{d-2} of the base-m-expansion of N. If a_{d-2} is not sufficiently small, go to the first step.
- Determine the complete base-m-expansion of N which gives an initial f_1 and set $f_2 = x - m$. Optimize this pair as explained above by changing the skewness, translating, and adding multiples of f_2 to f_1. If the coefficients of f_1 are too big, go to the first step.

- Using a sieve identify those f_1+cf_2 which have good root properties, where c is a polynomial of small degree with bounded coefficients, and output these pairs $(f_1 + cf_2, f_2)$. Go to the first step.

This algorithm outputs a lot of polynomial pairs which can be ranked as described above.

In the next sections we will describe an improvement of the first two steps of the algorithm above. The optimization step and the sieving step will not be affected.

2. Nonmonic linear polynomials

In this section we consider a substitute for the base-m-expansion in the case of a nonmonic polynomial f_2. Denote the linear polynomial by $f_2(x) = px - m$ and assume that p and m are coprime. We want to find a polynomial $f_1 = \sum_{i=0}^{d} a_i x^i$ of degree d such that $f_1(\frac{m}{p}) \cdot p^d = N$ holds, and the coefficients of f_1 should be as small as possible. As in the method described above we assume that the leading coefficient a_d is given. If the congruence

$$a_d m^d \equiv N \pmod{p} \tag{2.1}$$

does not hold, no polynomial f_1 satisfying $f_1(\frac{m}{p}) \cdot p^d = N$ exists.

Lemma 2.1. *Let N, d, a_d, p, and m be given such that $N \equiv a_d m^d \pmod{p}$ holds. Define $\tilde{m} := \sqrt[d]{\frac{N}{a_d}}$ and assume $m \geq \tilde{m}$. Then there exists a polynomial $f_1(x) = \sum_{i=0}^{d} a_i x^i$ satisfying*

- $f_1(\frac{m}{p}) \cdot p^d = N$,
- $|a_{d-1}| < p + d a_d \frac{m-\tilde{m}}{p}$, *and*
- $|a_i| < p + m$ *for* $0 \leq i \leq d-2$.

Proof. Let $r_d = N$ and choose successively for $i = d-1, \ldots, 0$

- $r_i = \dfrac{r_{i+1} - a_{i+1} m^{i+1}}{p}$ and
- $a_i = \frac{r_i}{m^i} + \delta_i$ with $0 \leq \delta_i < p$ such that $r_i \equiv a_i m^i \pmod{p}$ holds.

Then the r_i satisfy $N = a_d m^d + \cdots + a_{i+1} m^{i+1} p^{d-i-1} + r_i p^{d-i}$ or

$$r_i = \sum_{j=0}^{i} a_j m^j p^{i-j} \quad \text{for } i = 0, \ldots, d.$$

So we will get a polynomial satisfying the first property.

The second property follows from

$$|r_{d-1}| = \frac{1}{p}|N - a_d m^d| = \frac{a_d}{p}(m^d - \tilde{m}^d) < \frac{a_d}{p}(m - \tilde{m}) d m^{d-1}$$

and the definition of a_{d-1}.

For showing the last property we use

$$|r_{i-1}| = \frac{1}{p}|r_i - a_i m^i| = \frac{1}{p}\delta_i m^i < m^i$$

and the definition of a_i. □

The lemma above allows us to extend the first part of the Montgomery-Murphy polynomial selection algorithm to nonmonic linear polynomials. We choose a_d and p, solve the congruence (2.1), and, for each solution m, we compute the (first 3 coefficients of the) polynomial f_1 examining it more closely if a_{d-2} is sufficiently small. After that we go to the next pair (a_d, p).

This will not speed up the algorithm, in fact it will slow it down a bit since the polynomial expansion is now more expensive. But we have an overwhelming number of triples (a_d, p, m) at our disposal so that we can impose further restrictions on them in order to speed up the polynomial expansion. This will be done in the next section.

3. The improvement

We begin with a discussion of measuring the size of the polynomial f_1. We will work with the sup-norm of polynomials which is defined as follows:

Definition 3.1. Let $f(x) = \sum_{i=0}^{d} a_i x^i \in \mathbb{R}[x]$ be a polynomial of degree d and s a positive real number (skewness). We define

$$\sup(f, s) = \max_i |a_i s^{i-\frac{d}{2}}|$$

and

$$\sup(f) = \min_{s>0} \sup(f, s).$$

The (An) s for which the minimum is attained will be called optimal skewness.

Remark 3.2. We can also define the L^2-norm by

$$\sup_{L^2}(f, s) = \sqrt{\int_0^1 \int_0^1 \left(f\left(\frac{sx}{y}\right) \cdot \left(\frac{y}{\sqrt{s}}\right)^d\right)^2 dxdy}$$

and

$$\sup_{L^2}(f) = \min_{s>0} \sup_{L^2}(f, s).$$

This seems to give better estimates for the size properties of a polynomial, but we do not know how to use it in the following algorithm. We can at least bound the quotient of the two norms by constants (for fixed degree d).

From now on we assume $d \geq 4$. This is reasonable since at the crossover point of MPQS and GNFS degree 4 polynomials are optimal. It is also possible to carry over the algorithm below (with some modifications) to the case $d = 3$ and even $d = 2$. For the case $d = 4$, see also Remark 3.8.

Below we will present an algorithm for finding polynomials whose first three coefficients a_d, a_{d-1}, a_{d-2} are below some bounds $a_{d,\max}$, $a_{d-1,\max}$, resp. $a_{d-2,\max}$. It is possible to use these three bounds as input for the algorithm. However, in practice the following approach seems to be preferable.

Let $M < \sqrt[d+1]{N}$ be a bound on the sup-norm of the polynomials we want to find. For a given a_d let $\tilde{m} = \sqrt[d]{\frac{N}{a_d}}$ as above. We will allow $|a_0|$ and $|a_1|$ to be of size $\tilde{m}$. For an upper bound on the skewness we immediately get $s \leq \left(\frac{M}{a_d}\right)^{\frac{2}{d}}$. To get a lower bound we assume that in the polynomial expansion of Lemma 2.1 the worst

case happens for the first coefficient, namely $|a_1| = \tilde{m}$. With this assumption we obtain

$$s_{\min} = \left(\frac{\tilde{m}}{M}\right)^{\frac{2}{d-2}} \le s \le s_{\max} = \left(\frac{M}{a_d}\right)^{\frac{2}{d}} \tag{3.1}$$

for the optimal skewness. This immediately yields the bound

$$a_d \le \left(\frac{M^{2d-2}}{N}\right)^{\frac{1}{d-3}}$$

for a_d. We also get the bounds $|a_i| \le M s_{\min}^{\frac{d}{2}-i} =: a_{i,\max}$ for $\frac{d}{2} \le i < d$, and $|a_i| \le M s_{\max}^{\frac{d}{2}-i} =: a_{i,\max}$ for $i < \frac{d}{2}$ which depend on a_d.

We now assume that m is chosen near $\tilde{m}$ and that $p \le a_{d-1,\max}$ and $p \ll m$ holds. Then the polynomial expansion of Lemma 2.1 implies that the coefficient a_{d-1} is of order a_d and so is within the bounds. The other coefficients are of size $\tilde{m}$, so a_1 and a_0 are also within the bounds. Our task is to get the coefficients $a_{d-2}, \ldots, a_2$ sufficiently small. In the following we will show how to quickly estimate the size of a_{d-2}.

We now choose $p = \prod_{i=1}^{l} p_i$, where $p_i \equiv 1 \pmod{d}$ are (small) primes, $(p, a_d N) = 1$, and $p \le a_{d-1,\max}$. This choice implies that the equation $N \equiv a_d x^d \pmod{p}$ has either no solution or d^l solutions. In the latter case these can be written as

$$x_\mu = \sum_{i=1}^{l} x_{i,\mu_i}, \tag{3.2}$$

where $\mu = (\mu_1, \ldots, \mu_l)$ with $\mu_i \in \{1, \ldots, d\}$, $0 \le x_{i,\mu_i} < p$, $\frac{p}{p_i} \mid x_{i,\mu_i}$, and $\{x_{i,j} \bmod p_i \mid j = 1, \ldots, d\}$ are the d solutions of $N \equiv a_d x^d \pmod{p_i}$.

Remark 3.3. From each of these d^l solutions x_μ we will construct a polynomial pair via Lemma 2.1. The corresponding variables will get an additional subscript μ (e.g., a_{d-1} becomes $a_{d-1,\mu}$). We will represent some of these variables in a form such as (3.2), since in this form the d^l variables on the left-hand side are linear combinations of the ld variables on the right-hand side.

Let m_0 be the smallest integer bigger than $\tilde{m}$ and divisible by p and let

$$m_{i,j} = \begin{cases} m_0 + x_{i,j}, & i = 1, \\ x_{i,j}, & i > 1. \end{cases} \tag{3.3}$$

Then $m_\mu = \sum_{i=1}^{l} m_{i,\mu_i} = m_0 + x_\mu$ are d^l solutions of $N \equiv a_d x^d \pmod{p}$ near $\tilde{m}$.

Lemma 3.4. *Notations as above. There exist integers $0 \le e_{i,j} < p$ where $1 \le i \le l$ and $1 \le j \le d$ (see formula (3.6)) such that*

$$a_{d-1,\mu} = \sum_{i=1}^{l} e_{i,\mu_i} \tag{3.4}$$

satisfies

$$a_{d-1,\mu} m_\mu^{d-1} \equiv \frac{N - a_d m_\mu^d}{p} \pmod{p}. \tag{3.5}$$

Hence $a_{d-1,\mu}$ can be used in the expansion of N for the pair (p, m_μ).

Proof. First note that given $a_{d-1,\mu}$ modulo p for all μ by equation (3.5), it is sufficient to solve (3.4) modulo p since we can reduce the $e_{i,j}$ modulo p to satisfy $0 \le e_{i,j} < p$. Note also that the $e_{i,j}$ are not uniquely determined, since we can add a constant to all $e_{i,j}$, i fixed, $1 \le j \le l$ (reducing modulo p if necessary) and subtract the same constant from all $e_{i',j}$, i' fixed, $1 \le j \le l$.

Fix a $1 \le i \le l$, and let $\mu = (\mu_1, \dots, \mu_l)$, $\mu = (\mu'_1, \dots, \mu'_l)$ with $\mu_j = \mu'_j$ for $j \ne i$. We will show that $a_{d-1,\mu} - a_{d-1,\mu'} \bmod p$ does not depend on μ_j for $j \ne i$, but only on μ_i and μ'_i. This allows us to set

$$\begin{aligned} e_{1,1} &\equiv a_{d-1,(j,1,\dots,1)} \pmod p, \\ e_{i,1} &= 0 \qquad \text{for } i > 1 \text{ and} \\ e_{i,j} &\equiv a_{d-1,(1,\dots,1,j,1,\dots,1)} - a_{d-1,(1,\dots,1)} \pmod p \qquad \text{for } i > 1, j > 1 \end{aligned} \tag{3.6}$$

(in the last line j appears at the ith place). Because of the independence these $e_{i,j}$ satisfy (3.4).

Let $\tilde\mu = (\tilde\mu_1, \dots, \tilde\mu_l)$ and $\tilde{\mu'} = (\tilde{\mu'}_1, \dots, \tilde{\mu'}_l)$ be another pair with $\tilde\mu_j = \tilde{\mu'}_j$ for $j \ne i$ and $\tilde\mu_i = \mu_i$, $\tilde{\mu'}_i = \mu'_i$ (omitting or adding a $'$ means that everything outside the ith place remains constant; a $\tilde{}$ means that the ith place stays constant). We have to prove

$$a_{d-1,\mu} - a_{d-1,\mu'} \equiv a_{d-1,\tilde\mu} - a_{d-1,\tilde{\mu'}} \pmod{p_k},\ 1 \le k \le l. \tag{3.7}$$

We now multiply (3.5) by $\frac{1}{N} a_d m_\mu \bmod p$ and get

$$a_{d-1,\mu} \equiv \frac{1}{N} a_d m_\mu \frac{N - a_d m_\mu^d}{p} \pmod p.$$

For $k \ne i$ we have $m_\mu - m_{\mu'} = x_\mu - x_{\mu'} = x_{i,\mu_i} - x_{i,\mu'_i} \equiv 0 \pmod{p_k}$, whence we get

$$\begin{aligned} a_{d-1,\mu} - a_{d-1,\mu'} &\equiv \frac{a_d}{N} m_\mu \frac{a_d m_{\mu'}^d - a_d m_\mu^d}{p} \equiv \frac{a_d}{N} m_\mu \frac{a_d d m_\mu^{d-1}}{p_i} \frac{x_{i,\mu'_i} - x_{i,\mu_i}}{\frac{p}{p_i}} \\ &\equiv \frac{a_d d}{p_i} \frac{x_{i,\mu'_i} - x_{i,\mu_i}}{\frac{p}{p_i}} \pmod{\frac{p}{p_i}}, \end{aligned}$$

proving (3.7) for $k \ne i$.

For showing it modulo p_i we note that $m_\mu \equiv m_{\tilde\mu} \pmod{p_i}$ and get (as above)

$$a_{d-1,\mu} - a_{d-1,\tilde\mu} \equiv \frac{a_d}{N} m_\mu \frac{a_d m_{\tilde\mu}^d - a_d m_\mu^d}{p} \equiv \frac{a_d d (m_{\tilde\mu} - m_\mu)}{p} \pmod{p_i}$$

and analogously for $a_{d-1,\mu'} - a_{d-1,\tilde{\mu'}}$. Therefore

$$a_{d-1,\mu} - a_{d-1,\mu'} - a_{d-1,\tilde\mu} + a_{d-1,\tilde{\mu'}} \equiv \frac{a_d d (m_{\tilde\mu} - m_\mu - m_{\tilde{\mu'}} + m_{\mu'})}{p} \equiv 0 \pmod{p_i},$$

which completes the proof. □

Now we consider the next coefficient in the polynomial expansion corresponding to m_μ, and want to estimate its size. We use the approximation $m_\mu \approx m_0$ because m_0 is much bigger than p and m_μ differs from m_0 by at most lp. Since we are

free to add multiples of $(px - m_\mu)x^{d-2}$ to the polynomial expansion, we want that $\frac{a_{d-2,\mu}}{m_0}$ is very near to an integer. An approximation of this quantity is given by

$$\begin{aligned}\frac{a_{d-2,\mu}}{m_0} &\approx \frac{r_{d-2,\mu}}{m_0^{d-1}} = \frac{N - a_d m_\mu^d - a_{d-1,\mu} m_\mu^{d-1} p}{p^2 m_0^{d-1}} \\ &\approx \frac{N - a_d m_0^d - a_d d(m_\mu - m_0) m_0^{d-1} - a_{d-1,\mu} m_0^{d-1} p}{p^2 m_0^{d-1}} \\ &= \frac{N - a_d m_0^d}{p^2 m_0^{d-1}} + \frac{-a_d d(m_\mu - m_0) - a_{d-1,\mu} p}{p^2}.\end{aligned}$$

The transition from the first to the second line is done by using the binomial expansions of $(m_0 + (m_\mu - m_0))^a$, $a = d, d-1$, and omitting all monomials in the numerator for which the power of m_0 is less than $d-1$.

Definition 3.5. Let $f_0 = \frac{N - a_d m_0^d}{p^2 m_0^{d-1}}$ and for $1 \le i \le l$, $1 \le j \le d$,

$$f_{i,j} = -\frac{a_d d x_{i,j}}{p^2} - \frac{e_{i,j}}{p}.$$

With this definition the approximation above becomes

$$\frac{a_{d-2,\mu}}{m_0} \approx f_0 + \sum_{i=1}^{l} f_{i,\mu_i}.$$

The error made is $O(\frac{dl^2(da_d + p)}{m_0})$.

We now present an algorithm which, given an integer N and a degree $d \ge 4$, produces a list of polynomial pairs (f_1, f_2) with a common root modulo N such that the first three coefficients of f_1 are "small".

Algorithm 3.6. Input: a number N, a degree $d \ge 4$ of f_1, a bound M for the sup-norm of f_1 (or alternatively three bounds $a_{d,\max}$, $a_{d-1,\max}$ and $a_{d-2,\max}$ for the first three coefficients), a bound l on the minimal number of prime factors of p, and a bound p_b for these prime factors.

(1) Set $\mathcal{P} = \{r \equiv 1 \pmod d \mid r \text{ prime}, r \nmid N \text{ and } r < p_b\}$ and set $a_d = 0$.

(2) Increase a_d and terminate the algorithm if it exceeds $a_{d,\max} = \left(\frac{M^{2d-2}}{N}\right)^{\frac{1}{d-3}}$. Otherwise set

$$\mathcal{Q}(a_d) = \{r \in \mathcal{P} \mid \frac{a_d}{N} \not\equiv 0 \pmod r \text{ is a } d\text{th power modulo } r\}.$$

Also compute approximately $\tilde{m} = \sqrt[d]{\frac{N}{a_d}}$, $a_{d-1,\max} = \frac{M^2}{\tilde{m}}$ and $a_{d-2,\max} = \left(\frac{M^{2d-6}}{\tilde{m}^{d-4}}\right)^{\frac{1}{d-2}}$.

(3) For all subsets $\mathcal{P}'$ of at least l elements of $\mathcal{Q}(a_d)$ such that $p = \prod_{r \in \mathcal{P}'} r \le a_{d-1,\max}$ holds, execute the following three steps:
   (a) Compute $x_{i,j}$, $m_{i,j}$, and $e_{i,j}$ as in (3.2), (3.3), and (3.6), respectively.
   (b) Finally compute f_0 and $f_{i,j}$ as in Definition 3.5.
   (c) Set $\epsilon = \frac{a_{d-2,\max}}{m_0}$ and find those vectors μ for which

$$f_0 + \sum_{i=1}^{l} f_{i,\mu_i} \bmod \mathbb{Z} \in [-\epsilon, \epsilon]$$

holds and output the corresponding polynomial pair. This can be done by setting $l' = [\frac{l}{2}]$, computing the two lists $f_0 + \sum_{i=1}^{l'} f_{i,\mu_i} \bmod \mathbb{Z}$ and $-\sum_{i=l'+1}^{l} f_{i,\mu_i} \bmod \mathbb{Z}$, sorting these two lists, and checking each element of the second list to see whether it is in an ϵ-neighbourhood of an element of the first list.

Go to step (2).

Remark 3.7. In one pass of steps 3(a)–(c) we check d^l polynomial pairs in time $O(d^{\frac{l}{2}} \log d)$ resulting in a runtime of $O(d^{-\frac{l}{2}} \log d)$ per checked polynomial pair. So we try to choose l as large as possible.

Remark 3.8. For smaller numbers N (less than 105 digits, say) a polynomial pair of degree (4,1), i.e., $d = 4$, will be superior to one of degree (5,1). In this case the following modification produces better polynomial pairs. We no longer require that $|a_1|$ is of size $\tilde{m}$ which will restrict the degree of the polynomial c in the root sieve to 0, i.e., we search among $f_1 + c_0 f_2$, $c_0 \in \mathbb{Z}$, $|c_0|$ small for polynomials having good root properties. Then the bounds (3.1) will be replaced by

$$s_{\min} = \sqrt{\frac{\tilde{m}}{M}} \le s \le s_{\max} = \sqrt{\frac{M}{a_4}}$$

giving

$$|a_4| \le \left(\frac{M^8}{N}\right)^{\frac{1}{3}}, \quad |a_3| \le \left(\frac{M^{11}}{N}\right)^{\frac{1}{6}}, \quad \text{and} \quad |a_2| \le M.$$

The output of the algorithm above consists of polynomials such that all coefficients with the possible exception of a_1 are small enough. We then check whether $|a_1|$ also lies within the bounds (for a suitable skewness).

We now describe some variants of the algorithm above:

(1) Instead of considering every a_d we may only consider those which are divisible by a product of some small primes (60, say). This increases the root properties of the polynomial f_1 by adding projective roots modulo these small primes. On the other hand it reduces the number of available leading coefficients a_d which may force us to decrease the number l of prime factors in p which in turn slows down step 3(c) of the algorithm.

Alternatively it is also possible to identify good a_d using a sieve.

(2) For degree $d = 5$ and smaller l a large part of time is spent in the initialization of step 3(a). By also considering products p of the form $p = p_0 \prod_{i=1}^{l} p_i$, where p_0 is a number (not necessarily prime) such that $a_d x^d \equiv N \pmod{p_0}$ has exactly one solution, we can decrease the percentage of the initialization cost. For a set $\mathcal{P}'$ we first proceed as in parts 3(a)–(c) of the algorithm (this corresponds to $p_0 = 1$). Then we do these steps with other values of p_0 such that $p \le a_{d-1,\max}$ holds. For these values we can reuse some of the computations done for $p_0 = 1$.

(3) For very large N the number of admissible values for a_d is huge. We may decrease the sup-norm bound M, thereby shrinking the admissible a_d-interval, but we risk getting no polynomial satisfying this reduced sup-norm bound. So we can only restrict the search interval by selecting some of the a_d. Since we want to have the number l of different prime factors of p as large as possible (for very large N step 3(c) dominates the runtime), we reverse

the roles of p and a_d in the following way: select l primes out of $\mathcal{P}$ whose product p is smaller than $a_{d-1,\max}$. Now a_d must be congruent to N times a dth power modulo each of the l primes dividing p, and it has to satisfy a restriction on its size. So we get another knapsack problem whose solutions give pairs (a_d, p).

We may include informations modulo 60 into the knapsack problem to get only those a_d which are divisible by 60. As in the previous variant we may also use an auxiliary factor p_0, but since the initialization costs are not a problem, it is probably better to increase l.

4. Simple heuristic analysis

In this section we want to examine which quality we can expect using a given amount of time. This will only be a rough analysis not involving the root sieve and assuming that the number of examined polynomials is not too big.

As above let N be the integer to be factored, $d \geq 4$ the degree of the algebraic polynomial $f_1 = \sum_{i=0}^{d} a_i x^i$, and M a bound on its sup-norm. We want to search for polynomials whose sup-norm is less than M. Furthermore let $b < \frac{d-1}{2}$ be an integer. Using the algorithm described above we search for polynomials such that there exists a skewness $s \geq 1$ such that the following holds:

$$|a_i| \leq M s^{\frac{d}{2}-i} \quad \text{for } b \leq i \leq d \text{ and } \quad \sqrt[d]{\frac{N}{a_d}} \leq M s^{\frac{d}{2}-b}. \tag{4.1}$$

Since the coefficients $a_0, \ldots, a_{b-1}$ will be of size $\sqrt[d]{\frac{N}{a_d}}$, these conditions imply that the de-skewed sup-norm is at most M. The second condition implies that we can do a $(b+1)$-dimensional root sieve, i.e., using a sieve over polynomials $c \in \mathbb{Z}[x]$ of degree b with small coefficients, we search for polynomials $f_1 + cf_2$ having good root properties. In the previous section we only considered the case $b = 1$.

For a polynomial expansion we have by Lemma 2.1 $|a_{d-1}| \approx \max(p, a_d)$ and $|a_i| \approx \sqrt[d]{\frac{N}{a_d}}$ for $i = d-2, \ldots, 0$. Therefore we get the restriction $p \leq M s^{1-\frac{d}{2}}$. Then the average number W of polynomials we have to check in order to find one polynomial satisfying (4.1) is

$$W = \prod_{i=b+1}^{d-2} \max\left(1, \frac{\sqrt[d]{\frac{N}{a_d}}}{M s^{\frac{d}{2}-i}}\right).$$

These checks consist of a quick check of the size of a_{d-2} and for the

$$W' = \prod_{i=b+1}^{d-3} \max\left(1, \frac{\sqrt[d]{\frac{N}{a_d}}}{M s^{\frac{d}{2}-i}}\right)$$

survivors of this a slower check of the sizes of the remaining coefficients. Choosing p as a product of l primes, this can be done in time $O(\frac{W}{d^{\frac{l}{2}}}) + O(W')$. This is minimal if a_d is as large as possible, so we assume from now on that $a_d = M s^{-\frac{d}{2}}$. If we substitute this and multiply out, we get $W = N^{\cdots} M^{\cdots} s^z$, where $z \geq 0$ and $W' = N^{\cdots} M^{\cdots} s^{z'}$, where $z' \geq 0$ for $b \geq 1$. In order to minimize the work we choose s as small as possible for $b > 0$, i.e., $\sqrt[d]{\frac{N}{a_d}} = M s^{\frac{d}{2}-b}$, since a smaller s implies a

larger bound on p and therefore a larger l. In this case the second argument in the products above is always the maximum and we get

$$W = \left(\frac{N}{M^{d+1}}\right)^{\frac{(d-2-b)(d-1-b)}{d(d-1-2b)}} \quad \text{or}$$

$$M = N^{\frac{1}{d+1}} W^{-\frac{d(d-1-2b)}{(d+1)(d-2-b)(d-1-b)}}.$$

We now assume that the $O(\frac{W}{d^{\frac{l}{2}}})$-term dominates the $O(W')$-term. By checking one polynomial we obtain $M = N^{\frac{1}{d+1}}$ as expected. If we want to improve this by a factor f we have to check $f^{\frac{(d+1)(d-2-b)(d-1-b)}{d(d-1-2b)}}$ polynomials on average. For small values of d and b this exponent is tabulated in the following table:

b \ d	4	5	6	7
0	$\frac{5}{2}$	$\frac{18}{5}$	$\frac{14}{3}$	$\frac{40}{7}$
1	$\frac{5}{2}$	$\frac{18}{5}$	$\frac{14}{3}$	$\frac{40}{7}$
2	-	-	7	$\frac{48}{7}$

We note that this function takes the same values for $b = 0$ and $b = 1$, namely $\frac{(d+1)(d-2)}{d}$. So in these cases the amount of work for finding a polynomial with small sup-norm is equal, but for $b = 1$ we have a larger root sieve and can expect better root properties (always neglecting the $O(W')$-term).

5. Experimental results

This algorithm has been used for the polynomial selection stage of the factorization of many numbers. The largest number whose factorization has been completed and where a number of similar size has been factored using the original Montgomery-Murphy method is a composite 143-digit factor of $2^{1064} + 1$. We used the following parameters: a_5 ranged over all multiples of 60 between 1 and ca. $5.6 \cdot 10^9$, p was composed of 7 primes $\equiv 1 \pmod 5$ less than 1000 and an auxiliary factor less than 2^{15}. This took approximately 3 days on four 233 MHz Pentiums. Compared with the polynomial pair used for the factorization of the (slightly smaller) 143-digit composite $92! + 1$ which has been generated by the original Montgomery-Murphy method, the yield has been increased by 40%.

We also have done a short search for the 512-bit RSA challenge number factored in August 1999 ([1]). In this experiment p was composed of 7 primes $\equiv 1 \pmod 5$ and an auxiliary factor p_0. The range for the leading coefficient a_5 was $[1, 20000000]$, a_5 being a multiple of 60. Only for those polynomials whose de-skewed sup-norm was less than $2 \cdot 10^{23}$ a root sieve was performed. The best polynomial pair found was

$$\begin{aligned} f_1 = {} & 4985820x^5 + 15578368316860x^4 - 513748876280490487x^3 \\ & - 10211574130795357032973 44x^2 - 39893111467231678678251 49900x \\ & + 14658919460374074323550710377995600 \end{aligned}$$

and

$$f_2 = 87494555574829559x - 29394756538965034296055627 0613.$$

This polynomial pair has a yield approximately 32% higher than that of the polynomial pair used for the factorization. The time spent for the search was less than 9 hours on a 1 GHz Pentium.

For the 576-bit RSA challenge number factored in December 2003 ([2]), the algorithm presented in this article was intensely used. The best polynomial pair we found was

$$\begin{aligned} f_1 = {} & 46023405120x^5 + 10480176714921624x^4 - 293283243099549031036 03x^3 \\ & - 830838022743867257648284551x^2 \\ & + 261882930221985716573426822 1807627x \\ & - 231988217535862601582671892090396902425 \end{aligned}$$

and

$$f_2 = 59567272820312091 11x - 33291060200125678244148480384 3034,$$

and it was used for the factorization.

References

1. S. Cavallar, W. M. Lioen, H. J. J. teRiele, B. Dodson, A. K. Lenstra, P. L. Montgomery, B. Murphy et al., *Factorization of a 512-bit RSA modulus*, Report MAS-R0007, CWI.
2. J. Franke, T. Kleinjung et al., *RSA*-576, E-mail announcement, 2003. `http://www.crypto-world.com/announcements/rsa576.txt`
3. A. K. Lenstra and H. W. Lenstra, Jr. (eds.), *The Development of the Number Field Sieve*, Lecture Notes in Math. **1554**, Springer, 1993. MR1321217
4. B. A. Murphy and R. P. Brent, *On Quadratic Polynomials for the Number Field Sieve*, Computing Theory 98, ACSC **20**(3) (1998), pp. 199–215. MR1723947 (2000i:11189)
5. B. A. Murphy, *Modelling the Yield of Number Field Sieve Polynomials*, Algorithmic Number Theory - ANTS III, LNCS **1443** (1998), pp. 137–147. MR1726067 (2001d:11029)
6. B. A. Murphy, *Polynomial selection for the Number Field Sieve Integer Factorisation Algorithm*, Ph.D. thesis, The Australian National University, 1999.

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

E-mail address: `thor@math.uni-bonn.de`