MATHEMATICS OF COMPUTATION
Volume 84, Number 295, September 2015, Pages 2447–2457
S 0025-5718(2015)02926-3
Article electronically published on February 11, 2015

ROOT OPTIMIZATION OF POLYNOMIALS IN THE NUMBER FIELD SIEVE

SHI BAI, RICHARD P. BRENT, AND EMMANUEL THOMÉ

Abstract. The general number field sieve (GNFS) is the most efficient algorithm known for factoring large integers. It consists of several stages, the first one being polynomial selection. The quality of the chosen polynomials in polynomial selection can be modelled in terms of size and root properties. In this paper, we describe some algorithms for selecting polynomials with very good root properties.

1. The general number field sieve

The general number field sieve [14] is the most efficient algorithm known for factoring large integers. It consists of several stages including polynomial selection, sieving, filtering, linear algebra and finding square roots.

Let n be the integer to be factored. The number field sieve starts by choosing two irreducible and coprime polynomials $f(x)$ and $g(x)$ over $\mathbb{Z}$ which share a common root m modulo n. In practice, the notations $F(x, y)$ and $G(x, y)$ for the homogenized polynomials corresponding to f and g are often used. We want to find many coprime pairs $(a, b) \in \mathbb{Z}^2$ such that the polynomial values $F(a, b)$ and $G(a, b)$ are simultaneously smooth with respect to some upper bound B. An integer is smooth with respect to bound B (or B-smooth) if none of its prime factors are larger than B. Lattice sieving [19] and line sieving [6] are commonly used to identify such pairs (a, b). The running time of sieving depends on the quality of the chosen polynomials in polynomial selection, hence many polynomial pairs will be generated and optimized in order to produce a best one.

This paper discusses algorithms for root optimization in polynomial selection in the number field sieve. We mainly focus on polynomial selection with two polynomials, one of which is a linear polynomial.

2. Polynomial selection

For large integers, most polynomial selection methods [6, 11, 12, 16, 17] use a linear polynomial for $g(x)$ and a quintic or sextic polynomial for $f(x)$. Let $f(x) = \sum_{i=0}^{d} c_i x^i$ and $g(x) = m_2 x - m_1$. The standard method to generate such polynomial pairs is to expand n in base-(m_1, m_2) so $n = \sum_{i=0}^{d} c_i m_1^i m_2^{d-i}$.

The running time of sieving depends on the smoothness of the polynomial values $|F(a, b)|$ and $|G(a, b)|$. Let $\Psi(x, x^{1/u})$ be the number of $x^{1/u}$-smooth integers below x for some u. The Dickman-de Bruijn function $\rho(u)$ [9] is often used to estimate

Received by the editor June 14, 2013 and in revised form, October 30, 2013 and December 7, 2013.

2010 *Mathematics Subject Classification.* Primary 11Y05, 11Y16.

$\Psi(x, x^{1/u})$, since

$$\lim_{x\to\infty} \frac{\Psi(x, x^{1/u})}{x} = \rho(u).$$

The Dickman-de Bruijn function satisfies the differential equation

$$u\rho'(u) + \rho(u-1) = 0, \quad \rho(u) = 1 \text{ for } 0 \le u \le 1.$$

It can be shown that $\rho(u)$ satisfies the asymptotic estimate

$$\log(\rho(u)) = -(1 + o(1))\, u \log u \text{ as } u \to \infty.$$

For practical purposes, the frequency of smooth numbers can be approximated by the Canfield-Erdős-Pomerance theorem, which can be stated as follows [10].

Theorem 2.1. *For any fixed $\epsilon > 0$, we have*

$$\Psi(x, x^{1/u}) = xu^{-u(1+o(1))}$$

as $x^{1/u}$ and u tend to infinity, uniformly in the region $x \ge u^{u/(1-\epsilon)}$.

It is desirable that the polynomial pair can produce many smooth integers across the sieve region. Heuristically this requires that the size of polynomial values is small on average (Theorem 2.1). In addition, one can choose an algebraic polynomial $f(x)$ which has many roots modulo small prime powers. Such a choice is driven by inheritance of practices which already date back to the CFRAC era, where suitable multipliers were chosen precisely in order to optimize this very property [15,20]. Then the polynomial values are likely to be divisible by small prime powers. This may increase the smoothness probability for polynomial values. We first describe some methods [11,17] to estimate and compare the quality of polynomials.

2.1. **Sieving test.** A sieving experiment over short intervals is a relatively accurate method to compare polynomial pairs. It is often used to compare a few polynomial candidates in the final stage of the polynomial selection. Ekkelkamp [7] also described a method for predicting the number of relations needed in the sieving. The method conducts a short sieving test and simulates relations based on the test results. Experiments show that the prediction of the number of relations is close to the number of relations needed in the actual factorization.

2.2. **Size property.** Let (a, b) be pairs of coprime integers in the sieving region Ω. For the moment, we assume that a rectangular sieving region is used where $|a| \le K$ and $0 < b \le K$. We also assume that polynomial values $|F(a, b)|$ and $|G(a, b)|$ behave like random integers of similar size. The number of sieving reports (coprime pairs that lead to smooth polynomial values) can be approximated by

$$\frac{6}{\pi^2} \iint_{\Omega} \rho\left(\frac{\log|F(x,y)|}{\log B}\right) \rho\left(\frac{\log|G(x,y)|}{\log B}\right) \mathrm{d}x\, \mathrm{d}y.$$

The multiplier $6/\pi^2$ accounts for the probability of a, b being relatively prime.

Since G is a linear polynomial, we may assume that $\log(|G(a, b)|)$ does not vary much across the sieving region. A simplified approximation to compare polynomials (ignoring the constant multiplier) is to compare

$$\iint_{\Omega} \rho\left(\frac{\log|F(x,y)|}{\log B}\right) \mathrm{d}x\, \mathrm{d}y. \tag{2.1}$$

The base-(m_1, m_2) expansion [11, 12] gives polynomials whose coefficients are $O(n^{1/(d+1)})$. The leading coefficients c_d and c_{d-1} are much smaller than $n^{1/(d+1)}$. The coefficient c_{d-2} is slightly smaller than $n^{1/(d+1)}$. For such polynomials, it is often better to use a skewed sieving region where the sieving bounds for a, b have ratio $s \geq 1$, while keeping the area of the sieving region $2K^2$. The sieving bounds become $|a| \leq K\sqrt{s}$ and $0 < b \leq K/\sqrt{s}$. Each monomial in the polynomial is bounded by $c_i K^d s^{i-d/2}$.

In the integral (2.1), computing ρ is time-consuming (cf. [2]), especially if there are many candidates. We can use some coarser approximations. Since $\rho(u)$ is a decreasing function of u, we want to choose a polynomial pair such that the size of $|F(a,b)|$ and $|G(a,b)|$ is small on average over all (a,b). This roughly requires that the coefficients of the polynomials are small in absolute value. We can compare polynomials using the logarithm of an L^2-norm for the polynomial $F(x,y)$ by

$$\frac{1}{2}\log\left(s^{-d}\int_{-1}^{1}\int_{-1}^{1}F^2(xs,y)\,\mathrm{d}x\,\mathrm{d}y\right) \tag{2.2}$$

where s is the skewness of sieving region. Polynomials which minimize the expression (2.2) are expected to be better than others.

2.3. **Root property.** If a polynomial $f(x)$ has many roots modulo small primes and prime powers, the polynomial values may behave more smoothly than random integers of about the same size. Boender, Brent, Montgomery and Murphy [5,16–18] described some quantitative measures of this effect (root property).

Let p be a fixed prime. Let $\nu_p(x)$ denote the exponent of the largest power of p dividing the integer x and $\nu_p(0) = \infty$. Let S be a set of integers. We use (the same) notation $\nu_p(S)$ to denote the expected p-valuation of $x \in S$. If integers in S are random and uniformly distributed [1], the expected p-valuation $\nu_p(S)$ is

$$\nu_p(S) = \mathop{\mathrm{E}}_{x\in S}[\nu_p(x)] = \sum_{k=1}^{\infty}\Pr(\nu_p \geq k) = \sum_{k=1}^{\infty}\frac{1}{p^k} = \frac{1}{p-1}.$$

Thus, in an informal (logarithmic) sense, an integer x in S contains an expected power $p^{1/(p-1)}$.

Now, let S be a set of polynomial values $f(x)$. We use (the same) notation $\nu_p(S)$ (or $\nu_p(f)$) to denote the expected p-valuation of the polynomial values S. Hensel's lemma gives conditions when a root of $f \pmod{p^k}$ can be lifted to a root of $f \pmod{p^{k+1}}$.

Lemma 2.2 (Hensel's lemma)**.** *Let r_1 be a root of $f(x)$ modulo an odd prime p.*

1. *If r_1 is a simple root, $f(x) \pmod{p^k}$ has an unique root $r_k \equiv r_1 \pmod{p}$ for each $k > 1$.*
2. *If r_k is a multiple root [2] of $f(x) \pmod{p^k}$ for $k \geq 1$, there are two possible cases. If $p^{k+1} \mid f(r_k)$, then $\forall\, i \in [0,p)$, $p^{k+1} \mid f(r_k + i\,p^k)$. If $p^{k+1} \nmid f(r_k)$, r_k cannot be lifted to a root modulo p^{k+1}.*

Assume now that the integers x leading to the values $f(x) \in S$ are uniformly random. There are two cases. First, suppose $p \nmid \Delta(f)$, the discriminant of $f(x)$, then p is an unramified prime, and hence $f(x) \pmod{p}$ has only simple roots.

[1] We consider integer random variables within a large enough bounded sample space.

[2] Let r_k be a root of $f \pmod{p^k}$. We say that r_k is a *multiple* root of $f \pmod{p^k}$ if $f'(r_k) \equiv 0 \pmod{p}$; otherwise it is a *simple* root.

Let n_p be the number of roots for $f(x) \pmod p$. The expected p-valuation of polynomial values is $\nu_p(f) = n_p/(p-1)$ (apply the formula above, using $\Pr(\nu_p \geq k) = n_p/p^k$).

The second case is when $p \mid \Delta(f)$. Here one may get multiple roots. The expected p-valuation may be obtained by counting the number of lifted roots.

2.3.1. *Homogeneous polynomials.* In the number field sieve, we want to know the expected p-valuation of homogeneous polynomial values $F(a,b)$, where (a,b) is a pair of coprime integers, and $F(x,y)$ is the homogenous polynomial corresponding to $f(x)$. We assume in the following that (a,b) is a uniformly random pair of coprime integers. We have

$$\nu_p(F(a,b)) = \nu_p(F(\lambda a, \lambda b)) \tag{2.3}$$

for any integer λ coprime to p. A pair of coprime integers (a,b) maps to a point $(a:b)$ on the projective line $\mathbf{P}^1(\mathbb{F}_p)$. Because of property (2.3) above, pairs for which $\nu_p(F(a,b)) > 0$ correspond to the points of the zero-dimensional variety on $\mathbf{P}^1(\mathbb{F}_p)$ defined by the polynomial F.

The projective line $\mathbf{P}^1(\mathbb{F}_p)$ has $p+1$ points, consisting of p affine points which can be represented as $(x:1)$ with $x \in \mathbb{F}_p$, together with the point at infinity $(1:0)$. Among these, the zeroes of F correspond, for affine points $(x:1)$, to affine roots $x \in \mathbb{F}_p$ of the dehomogenized polynomial f. The point at infinity is a zero of F if and only if the leading coefficient c_d of f cancels modulo p. If F has a total of n_p affine and projective zeroes in $\mathbf{P}^1(\mathbb{F}_p)$, then $F(a,b)$ for coprime (a,b) is divisible by p with probability $n_p/(p+1)$.

It is also possible to look at (a,b) modulo a prime power p^k. Then (a,b) maps to an equivalence class $(a:b)$ on the projective line over the ring $\mathbb{Z}/p^k\mathbb{Z}$. The p-valuation of F at $(a:b) \in \mathbf{P}^1(\mathbb{Z}/p^k\mathbb{Z})$ (an integer between 0 and $k-1$, or "k or more") conveys the information of what happens modulo p^k. There are $p^k + p^{k-1}$ points in $\mathbf{P}^1(\mathbb{Z}/p^k\mathbb{Z})$ (p^k affine points of the form $(x:1)$, while the remaining p^{k-1} points at infinity are written as $(1:py)$). Therefore, a coprime pair (a,b) chosen at random maps to a given point in $\mathbf{P}^1(\mathbb{Z}/p^k\mathbb{Z})$ with probability $1/(p^{k-1}(p+1))$.

Given an unramified p, let $F(x,y) \pmod p$ have n_p affine and projective roots (zeroes on $\mathbf{P}^1(\mathbb{F}_p)$). In application of the Hensel Lemma (applied to f at an affine root x, or to $p^{d-1}f(\frac{1}{py})$ above the possible projective root), there is a constant number n_p of points $(a:b) = \mathbf{P}^1(\mathbb{Z}/p^k\mathbb{Z})$ such that $\nu_p(F(a:b)) \geq k$, as k grows. Thus, the expected p-valuation $\nu_p(F)$ is

$$\nu_p(F) = \sum_{k=1}^{\infty} \frac{n_p}{p^{k-1}(p+1)} = \frac{n_p p}{p^2-1}. \tag{2.4}$$

For ramified p, simply counting the number n_p of affine and projective roots modulo p is not sufficient to deduce $\nu_p(F)$. One can substitute n_p by n_{p^k} (the number of roots modulo p^k) in the above summation. For convenience, one can also define a truncated version where

$$\nu_p(F,e) = \sum_{k=1}^{e} \frac{n_{p^k}}{p^{k-1}(p+1)}. \tag{2.5}$$

Murphy [17, p. 49] defines the $\alpha(F,B)$ function to compare the cumulative expected p-valuation of polynomial values to random integers of similar size. The

function $\alpha(F, B)$ can be considered as the logarithmic benefit of using polynomial values compared to using random integers

$$\alpha(F, B) = \sum_{\substack{p \le B \\ p \text{ prime}}} \left(\frac{1}{p-1} - \nu_p(F) \right) \log p, \tag{2.6}$$

where the summand can be written as $\left(1 - \frac{n_p p}{p+1}\right) \frac{\log p}{p-1}$ when p is unramified. In the number field sieve, $\alpha(F, B)$ is often negative since we are interested in the case when $F(x, y)$ has on average more than one root modulo small primes.

2.4. **Steps in polynomial selection.** Polynomial selection can be divided into three steps: polynomial generation, size optimization and root optimization.

In the polynomial selection, we first generate some polynomials of relatively good size (cf. Subsection 2.2). Two efficient algorithms are given by Kleinjung [11, 12]. The size and root properties of these polynomials can then be further optimized using translation and rotation.

Translation of $f(x)$ and $g(x)$ by $t \in \mathbb{Z}/n\mathbb{Z}$ gives a new polynomial pair $f(x+t)$ and $g(x+t)$. The new common root is $m_1/m_2 - t \pmod{n}$. Translation only affects the size property. Rotation by a polynomial $\lambda(x)$ produces a new polynomial $f_{\lambda(x)}(x) = f(x) + \lambda(x)(m_2 x - m_1)$. The linear polynomial and common root is unchanged during rotation. $\lambda(x)$ is often a linear or quadratic polynomial, depending on the size of n and the skewness of $f(x)$. Rotation can affect both size and root properties.

Given a polynomial pair, translation and rotation are used to find a polynomial of smaller (skewed) norm (equation (2.2)). This is called size optimization. After size optimization, many polynomials can have comparable size. Given $f(x)$, we can use polynomial rotation to find a related polynomial $f_{\lambda(x)}(x)$ which has a much smaller α-value but similar size. This step is referred to as root optimization. If the skewness of the polynomial is large, the size property of the polynomial may not be altered significantly. As an indication of this, the skewed L^∞ norm of f, defined as $\max_i |s^{i-d/2} c_i|$, remains unchanged as long as the trailing coefficients of $f_\lambda(x)$ do not dominate. This is true for the polynomials generated by the algorithm of Kleinjung [12], where the skewness of the polynomials is likely to be large.

We discuss some algorithms for root optimization in the following sections. We will focus on rotation using linear polynomials. The idea naturally generalises to quadratic and higher degree rotations which are needed for large integers such as RSA-768 [13] and RSA-896.

3. Root sieve

We consider linear rotations defined by $f_{u,v}(x) = f(x) + (ux + v)g(x)$. We want to choose (u, v) such that $f_{u,v}(x)$ has a small α-value. The straightforward way is to look at individual polynomials $f_{u,v}(x)$ for all possible (u, v)'s and compare their α-values. This is time-consuming and impractical since the permissible bounds for u and v are often huge. Murphy [17, p. 84] describes a sieve-like procedure, namely the root sieve, to find polynomials with good root properties. We describe the root sieve in Algorithm 1.

Let B be the upper bound for primes in equation (2.6) and let U, V be bounds for the linear rotation such that $|u| \le U$ and $|v| \le V$. We often choose $V \approx sU$ (cf. Subsection 2.2). The root sieve fills an array with estimated α-values. The α-values

are estimated from p-valuations for primes $p \le B$. Alternatively, it is sufficient to calculate the summation of the weighted p-valuations $\nu_p(F)\log p$ for the purpose of comparison. The idea of the root sieve is that, when r is a root of $f_{u,v}(x) \pmod{p^k}$, it is also a root of $f_{u+ip^k,v+jp^k}(x) \pmod{p^k}$ for $i, j \in \mathbb{Z}$.

Algorithm 1: Murphy's root sieve

Input : a polynomial pair f, g; integers U, V, B;
Output: an array of approximated α-values of dimension $(2U+1)\times(2V+1)$;

```
for p ≤ B, p prime do
    for k where p^k ≤ B do
        for x ∈ [0, p^k − 1] do
            for u ∈ [0, p^k − 1] do
                compute v in f(x) + (ux + v)g(x) ≡ 0 (mod p^k);
                update ν_p(F_{u+ip^k, v+jp^k}) by sieving.
```

In general, the root sieve does not affect the projective roots significantly. It is sufficient to only consider the affine roots' contribution to the α-value. In the end, we identify good slots (those with small α-values) in the sieving array. For each slot (polynomial), we can compute a more accurate α-value with a large bound $\tilde{B} > B$ and re-optimize its size using translation only (which will not affect the root property).

With $B^2 \ll UV$, the asymptotic complexity of Murphy's root sieve is

$$\begin{aligned}
&\sum_{\substack{p\le B\\ p \text{ prime}}} \left(\sum_{k=1}^{\lfloor \frac{\log B}{\log p} \rfloor} p^k p^k \left(O(1) + \frac{(2U+1)(2V+1)}{p^{2k}} \right) \right) \\
&= O\left(\frac{B^3}{\log B}\right) + (2U+1)(2V+1) \sum_{\substack{p\le B\\ p \text{ prime}}} \left\lfloor \frac{\log B}{\log p} \right\rfloor \\
&\approx (2U+1)(2V+1)\log B \int_2^B \frac{1}{\log^2 p} dp \\
&= O\left(UV\frac{B}{\log B}\right).
\end{aligned}$$

We are interested in small primes and hence $B/\log B$ is small. The sieving bounds U, V dominate the running time $O(UVB/\log B)$.

4. A faster root sieve

In the root sieve, we compare the number of roots of polynomials $f_{u,v}(x)$ for small primes and prime powers. In most cases, the roots are simple. Hence their average p-valuation follows immediately from equation (2.4), and there is no need to count the lifted roots. We describe a faster root sieve (Algorithm 2) taking advantage of this idea.

First, we show that the cases corresponding to simple roots (cf. equation (2.4)) can be dealt with by a sieve. Suppose r_1 is a simple root of $f(x) \pmod{p}$. There exists a unique lifted root r_k of $f(x) \pmod{p^k}$ for each $k > 1$. In addition, each lifted root r_k is a simple root of $f(x) \pmod{p}$. Let r_k be a simple root of $f_{u,v}(x) \pmod{p^k}$ for some $k \ge 1$. It is clear that r_k is also a simple root of polynomials

$f_{u+ip^k,v+jp^k}(x) \pmod{p^k}$. Given a simple root r_1 of a polynomial $f_{u,v}(x) \pmod{p}$, the contribution of the root r_1 to $\nu_p(F_{u,v})$ is $p/(p^2-1)$. We can update this value[3] for all rotated polynomials $f_{u+ip,v+jp}(x)$ in a sieve (Line 8 of Algorithm 2).

Second, we consider the multiple roots. Let r_k be a multiple root of a rotated polynomial $f_{u,v}(x) \pmod{p^k}$. It is also a multiple root for rotated polynomials $f_{u+ip^k,v+jp^k}(x) \pmod{p^k}$. Hence we can also update the score in a sieve, but only for rotated polynomials $f_{u+ip^k,v+jp^k}(x) \pmod{p^k}$. The lifted roots of $f_{u+ip^k,v+jp^k}(x) \pmod{p^{k+1}}$ can have different behaviours (cf. Lemma 2.2). We need to lift to count the multiple roots (Line 10 of Algorithm 2).

Algorithm 2: A faster root sieve

input : a polynomial pair f, g; integers U, V, B;
output: an array of approximated α-values of dimension $(2U+1) \times (2V+1)$;

```
1  for p ≤ B, p prime do
2      for x ∈ [0, p − 1] do
3          compute ũ such that ũg²(x) ≡ f(x)g′(x) − f′(x)g(x) (mod p);
4          for u ∈ [0, p − 1] do
5              compute v such that f(x) + (ux + v)g(x) ≡ 0 (mod p);
6              if u ≠ ũ;
7              then
8                  update ν_p(F_{u+ip,v+jp}) in sieving.
9              else
10                 lift to count multiple roots of f_{ū,v̄}(x) (mod p^k) such that (ū, v̄) ≡ (u, v)
                   (mod p), ū, v̄ ≤ p^k ≤ B and then sieve;
```

Line 5 of Algorithm 2 describes the following optimization. Given some $r \in \mathbb{Z}/p\mathbb{Z}$, we want to know when r is a multiple root for some polynomial $f_{u,v}(x) \pmod{p}$. If $f(r) + (ur+v)g(r) \equiv 0 \pmod{p}$ and $f'(r) + ug(r) + (ur+v)g'(r) \equiv 0 \pmod{p}$, then we get

$$ug^2(r) \equiv f(r)g'(r) - f'(r)g(r) \pmod{p} \tag{4.1}$$

since $(ur+v) \equiv -f(r)/g(r) \pmod{p}$. Therefore, only one in p values of u admit a multiple root at $r \pmod{p}$. For the other u's, we can compute v and update the simple contribution $p/(p^2-1)$ in the sieve. If, however, r is a multiple root of $f_{u,v}(x) \pmod{p}$, we have to lift to count the multiple roots.

The asymptotic running time has the same magnitude as Murphy's root sieve where

$$\sum_{\substack{p \le B \\ p \text{ prime}}} \left(p \left((p-1) \frac{(2U+1)(2V+1)}{p^2} + O\left(\frac{UV}{p^2}\right)\right)\right) = O\left(UV \frac{B}{\log B}\right).$$

In practice, however, we benefit from this optimization which avoids most prime powers. For comparison, Murphy's root sieve takes about $4UV \sum_{p \le B} \lfloor \log B / \log p \rfloor$ operations, while Algorithm 2 takes about $4UV \sum_{p \le B} 1$ operations. Taking $B = 200$ for instance. $\sum_{p \le 200} \lfloor \log 200 / \log p \rfloor = 60$ and $\sum_{p \le 200} 1 = 46$. Thus the speedup is about 1.3.

[3] $\nu_p(F_{u,v}) \log p$, the contribution of the root $r_1 \pmod{p}$ to $\alpha(F_{u,v}, B)$.

5. A two-stage method

If the permissible rotation bounds U, V are large, the root sieve can take a long time for each polynomial. We give a two-stage algorithm for the root optimization. The algorithm is motivated by previous work by Gower [8], Papadopoulos (personal communication), Stahlke and Kleinjung [21], who suggested to consider congruence classes modulo small primes.

The root optimization is based on the following ideas. A polynomial with only a few roots modulo small prime powers is less likely to have a small α-value. Therefore, rotated polynomials with many roots modulo small prime powers $p_i^{e_i}$ are first detected, with $p_i \le P$ for some tiny bound $P \ll B$. How exactly the powers e_i are chosen is discussed at the end of this section. A further root sieve (cf. Algorithm 2) for larger prime powers $p_i^{e_i}$ (where $P < p_i \le B$ and $p_i^{e_i} \le B$) can then be applied.

For convenience, let primes p_i be ordered such that $p_i \le p_j$ when $i \le j$.

5.1. **Stage 1.** Let $p_1^{e_1}, \ldots, p_s^{e_s}$ be the distinct prime powers and such that $p_s \le P$. Let $M = \prod_{i=1}^{s} p_i^{e_i}$. In the first stage, we find some rotated polynomial $f_{u_0,v_0}(x)$ which has the smallest approximated α-value (equations (2.5) and (2.6)) among all $u, v \in \mathbb{Z}/M\mathbb{Z}$.

Gower [8] described an algorithm to find such $f_{u_0,v_0}(x)$. The method first fixes some root set $\{r_{i,j}\}$ modulo $p_i^{e_i}$ and then finds the rotated coefficients $(u_{0,i}, v_{0,i})$ (mod $p_i^{e_i}$). Finally, the CRT (Chinese Remainder Theorem) is applied to recover (u_0, v_0) (mod M). However, it is not guaranteed that such (u_0, v_0) leads to the smallest α-value among all $u, v \in \mathbb{Z}/M\mathbb{Z}$ given the root set being fixed in advance. We describe a better method based on the lifting idea in Section 4.

Let $p_i^{e_i}$ be fixed. We first find some polynomial $f_{u_{0,i},v_{0,i}}(x)$ that has good[4] approximated p_i-valuation $\nu_{p_i}(F_{u,v}, e_i)$ (cf. equation (2.5)) among all $u, v \in \mathbb{Z}/p_i^{e_i}\mathbb{Z}$. We use the lifting method in a p_i^2-ary tree (of height e_i) where each node represents some rotated polynomial $f_{u,v}$ modulo some $\tilde{e}_i \le e_i$. A depth-search method can be used since the bottom level leaves of the tree are most interesting. The number of nodes in this p_i^2-ary tree (of height e_i) can be bounded above by $p_i^{e_i}$ using the relation in equation (4.1). Given polynomials $f_{u_{0,i},v_{0,i}}(x)$ (mod $p_i^{e_i}$) for all $1 \le i \le s$, we can then use the CRT to recover a set of good pairs $\{(u_0, v_0)$ (mod $M)\}$.

5.2. **Stage 2.** Fix some (u_0, v_0) (mod M). We apply the root sieve on the sublattice defined by $(u_0 + \gamma M, v_0 + \beta M)$ where $(\gamma, \beta) \in \mathbb{Z}^2$. The points on the sublattice are expected to give rotated polynomials with promising root properties, since the polynomials are constructed to have many roots modulo M.

In Stage 1, we often choose the p_i's to be the smallest consecutive primes since they are likely to contribute most to the α-value. The exponents e_i can be chosen such that $M \approx U$ in practice. Note $U \ll V$ if s is large. Since $u_0 \approx M \approx U$, it is sufficient to sieve a single line in Stage 2.

Compared to a full root sieve over $\mathbb{Z}^2$, the search space is reduced by a factor of M^2. The root sieve in Stage 2 runs asymptotically in $UV(B-P)/(M^2 \log(B-P)) \approx (V/M)(B/\log B)$ as $B \ll V$ and $P \ll B$.

[4] In practice, one can record the top l such polynomials in a priority queue.

Remark 5.1. In Stage 2, the polynomials $\bar{f}$ not on the sublattice are discarded since we assume that they are unlikely to give rise to polynomials with good root properties: the sum of the approximated p_i-valuations $\sum_{i=1}^{s} \nu_{p_i}(\bar{F}, e_i) \log p_i$ is smaller than $\sum_{i=1}^{s} \nu_{p_i}(F, e_i) \log p_i$. However, it may be possible that $\sum_{i=1}^{s} \nu_{p_i}(\bar{F}) \log p_i \geq \sum_{i=1}^{s} \nu_{p_i}(F) \log p_i$ (for the exact p_i-valuations). Here we assume that $\nu_{p_i}(F) \approx \nu_{p_i}(F, e_i)$ which is plausible if the number of lifted multiple roots becomes stationary for $p_i^k \geq p_i^{e_i}$ (but also see below in 5.3).

5.3. **Choice of the valuations e_i.** There are several approaches to choosing the valuations e_i in Stage 1. To start with, for consistency with the rest of the procedure, it is natural to restrict to $p_i^{e_i} \leq B$, whence $e_i \leq \lfloor \log_{p_i} B \rfloor$. It is also reasonable to require that polynomials $f_{u,v}$ considered in Stage 2, for u, v within the lattice $(u_0 + \gamma M, v_0 + \beta M)$, all share the same lifting patterns for their p_i-adic roots (in terms of numbers of roots and multiplicities as lifting proceeds), which is to say that the number of lifted roots for these polynomials should be stationary above $p_i^{e_i}$. This would imply that we set e_i to at most $\nu_{p_i}(\operatorname{disc} f_{u,v})$, since setting to a larger value would not differentiate between lifting patterns. Of course the dependency of the latter expression on u, v is cumbersome. An upper bound would be needed, and can be obtained with moderate efforts by considering $\operatorname{disc} f_{u,v}$ as a bivariate polynomial in u, v. In practice, however, it is generally satisfactory to restrict to the bound $\nu_{p_i}(\operatorname{disc} f)$ as a first guess.

Another aspect leads to consider the maximum values for e_i in a more relaxed way. So far, in our improved root sieve, we have ignored the size property of polynomials in the algorithms. In practice, we may want to tune the parameters by trying several sets of parameters (varying p_i's and e_i in Stage 1). We can run a test root sieve in short intervals. The set of parameters which generates the best score (considering both size and root properties) is then used. To summarize, while a bound such as $e_i = \min(\nu_{p_i}(\operatorname{disc} f), \lfloor \log_{p_i} B \rfloor)$ would be advised by the analysis, efficiency considerations lead us to consider several sets of parameters around this value. This compensates for the acknowledged inaccuracy in considering $\nu_{p_i}(\operatorname{disc} f)$.

6. Conclusion

Root optimization aims to produce polynomials that have many roots modulo small primes and prime powers. We gave some faster methods for root optimization based on Hensel's lifting lemma and root sieve on congruence classes modulo small prime powers. The algorithms described here have been implemented in the software CADO-NFS [3] and tested in practice (e.g. for the factorization of the 704-bit RSA challenge [4]).

Acknowledgements

The authors are grateful to Pierrick Gaudry, Guillaume Hanrot, Thorsten Kleinjung, Jason Papadopoulos and Paul Zimmermann for many helpful comments and discussions on drafts of this paper. The authors would also like to thank an anonymous referee for valuable comments and suggestions.

The work of the first author was carried out at the Australian National University (ANU). He would like to thank the Research School of Computer Science (ANU) for funding and the Mathematical Sciences Institute (ANU) for providing computing

facilities (ORAC) and support. He would also like to thank NeSI (New Zealand eScience Infrastructure) and the Centre for eResearch at the University of Auckland for providing computing facilities and support.

References

[1] K. Aoki and H. Ueda, *Sieving using bucket sort*, Advances in cryptology—ASIACRYPT 2004, Lecture Notes in Comput. Sci., vol. 3329, Springer, Berlin, 2004, pp. 92–102, DOI 10.1007/978-3-540-30539-2_8. MR2150089 (2006c:11147)

[2] E. Bach and R. Peralta, *Asymptotic semismoothness probabilities*, Math. Comp. **65** (1996), no. 216, 1701–1715, DOI 10.1090/S0025-5718-96-00775-2. MR1370848 (98a:11123)

[3] S. Bai, C. Bouvier, A. Filbois, P. Gaudry, L. Imbert, A. Kruppa, F. Morain, E. Thomé, P. Zimmermann, `CADO-NFS`, an implementation of the number field sieve algorithm. Release `2.0`, available from `http://cado-nfs.gforge.inria.fr`, 2013.

[4] S. Bai, E. Thomé, P. Zimmermann, Factorisation of RSA-704 with CADO-NFS. Report, 2012. `http://eprint.iacr.org/2012/369.pdf`.

[5] H. Boender, *Factoring large integers with the quadratic sieve*. PhD thesis, Leiden University, 1997.

[6] J. P. Buhler, H. W. Lenstra Jr., and C. Pomerance, *Factoring integers with the number field sieve*, The development of the number field sieve, Lecture Notes in Math., vol. 1554, Springer, Berlin, 1993, pp. 50–94, DOI 10.1007/BFb0091539. MR1321221

[7] W. Ekkelkamp, *Predicting the sieving effort for the number field sieve*, Algorithmic number theory, Lecture Notes in Comput. Sci., vol. 5011, Springer, Berlin, 2008, pp. 167–179, DOI 10.1007/978-3-540-79456-1_11. MR2467845 (2009k:11203)

[8] J. E. Gower, *Rotations and translations of number field sieve polynomials*, Advances in cryptology—ASIACRYPT 2003, Lecture Notes in Comput. Sci., vol. 2894, Springer, Berlin, 2003, pp. 302–310, DOI 10.1007/978-3-540-40061-5_18. MR2093587 (2005f:11038)

[9] A. Granville, *Smooth numbers: computational number theory and beyond*, Algorithmic number theory: lattices, number fields, curves and cryptography, Math. Sci. Res. Inst. Publ., vol. 44, Cambridge Univ. Press, Cambridge, 2008, pp. 267–323. MR2467549 (2010g:11214)

[10] A. Hildebrand and G. Tenenbaum, *Integers without large prime factors*, J. Théor. Nombres Bordeaux **5** (1993), no. 2, 411–484. MR1265913 (95d:11116)

[11] T. Kleinjung, *On polynomial selection for the general number field sieve*, Math. Comp. **75** (2006), no. 256, 2037–2047 (electronic), DOI 10.1090/S0025-5718-06-01870-9. MR2249770 (2007f:11140)

[12] T. Kleinjung, Polynomial selection. In *CADO workshop on integer factorization*, INRIA Nancy, 2008. `http://cado.gforge.inria.fr/workshop/slides/kleinjung.pdf`.

[13] T. Kleinjung, K. Aoki, J. Franke, A. K. Lenstra, E. Thomé, J. W. Bos, P. Gaudry, A. Kruppa, P. L. Montgomery, D. A. Osvik, H. te Riele, A. Timofeev, and P. Zimmermann, *Factorization of a 768-bit RSA modulus*, Advances in cryptology—CRYPTO 2010, Lecture Notes in Comput. Sci., vol. 6223, Springer, Berlin, 2010, pp. 333–350, DOI 10.1007/978-3-642-14623-7_18. MR2725602

[14] A. K. Lenstra and H. W. Lenstra, Jr., editors, *The Development of the Number Field Sieve*, volume 1554, Lecture Notes in Mathematics, Springer, 1993.

[15] M. A. Morrison and J. Brillhart, *A method of factoring and the factorization of F_7*, Math. Comp. **29** (1975), 183–205. MR0371800 (51 #8017)

[16] B. Murphy, *Modelling the yield of number field sieve polynomials*, Algorithmic number theory (Portland, OR, 1998), Lecture Notes in Comput. Sci., vol. 1423, Springer, Berlin, 1998, pp. 137–150, DOI 10.1007/BFb0054858. MR1726067 (2001d:11029)

[17] B. A. Murphy, *Polynomial selection for the number field sieve integer factorisation algorithm*, PhD thesis, The Australian National University, 1999.

[18] B. Murphy and R. P. Brent, *On quadratic polynomials for the number field sieve*, Computing theory '98 (Perth), Aust. Comput. Sci. Commun., vol. 20, Springer, Singapore, 1998, pp. 199–213. MR1723947 (2000i:11189)

[19] J. M. Pollard, *The lattice sieve*, The development of the number field sieve, Lecture Notes in Math., vol. 1554, Springer, Berlin, 1993, pp. 43–49, DOI 10.1007/BFb0091538. MR1321220

[20] C. Pomerance and S. S. Wagstaff Jr., *Implementation of the continued fraction integer factoring algorithm*, Congr. Numer. **37** (1983), 99–118. MR703581 (85c:11124)
[21] C. Stahlke and T. Kleinjung, *Ideas for Finding Better Polynomials to Use in GNFS*. In Workshop on Factoring Large Numbers, Discrete Logarithms and Cryptanalytical Hardware, Institut für Experimentelle Mathematik, Universität Duisburg-Essen, 2008.

Department of Mathematics, University of Auckland, Auckland, New Zealand.
E-mail address: shih.bai@gmail.com

Mathematical Sciences Institute, Australian National University, Australia.
E-mail address: nfs@rpbrent.com

INRIA Nancy, Villers-lès-Nancy, France.
E-mail address: emmanuel.thome@inria.fr