SIZE OPTIMIZATION OF SEXTIC POLYNOMIALS IN THE NUMBER FIELD SIEVE

SHI BAI, CYRIL BOUVIER, ALEX KRUPPA, AND PAUL ZIMMERMANN

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. We describe some methods to optimize the size property of sextic polynomials.

1. Introduction to GNFS

The general number field sieve [12] is the most efficient algorithm known for factoring large integers. It has been used in many (current and previous) record factorizations such as RSA-768 [19] and RSA-704 [2]. GNFS consists of several stages including polynomial selection, sieving, filtering, linear algebra and finding square roots.

Let n be the integer to be factored. In polynomial selection, we want to choose two irreducible and coprime polynomials $f(x)$ and $g(x)$ over $\mathbb{Z}$ which share a common root m modulo n. In practice, the homogenized polynomials $F(x,y)$ and $G(x,y)$ are often used. We want to find many coprime pairs $(a,b) \in \mathbb{Z}^2$ such that the polynomials values $F(a,b)$ and $G(a,b)$ are simultaneously smooth with respect to some bounds B_1 and B_2. An integer is smooth with respect to bound B (or B-smooth) if none of its prime factors are larger than B. The line sieving and lattice sieving [18] 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 good one.

This paper discusses algorithms for size optimization in polynomial selection in the number field sieve. We focus on polynomial selection with two polynomials, one of which is a linear polynomial and the other is a polynomial of degree six. Such polynomials are of great practical interest since they have been used in current and previous record factorizations such as RSA-768 [19] and may be used for future records.

2. Polynomial selection

For large integers, most methods for polynomial selection [5, 10, 11, 13, 14] in GNFS use a linear polynomial for $g(x)$ and a quintic or sextic polynomial for $f(x)$. 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 polynomial pair is given by $f(x) = \sum_{i=0}^{d} c_i x^i$ and $g(x) = m_2 x - m_1$.

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 $u > 0$. The Dickman-de Bruijn function $\rho(u)$ [8] is often used to estimate the density of smooth numbers $\Psi(x, x^{1/u})$. It can be shown that

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

The Dickman-de Bruijn function satisfies the differential-difference equation

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

It may be shown that ρ 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 (Corollary 1.3 from [9]).

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

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 $x \geq u^{u/(1-\epsilon)}$.

We want to choose the polynomials in a way such that it can produce many smooth polynomial values across the sieve region. This heuristically requires that the size of polynomial values is small in general. In addition, one can choose an algebraic polynomial $f(x)$ which has many roots modulo small prime powers. Then the polynomial values are likely to be divisible by small prime powers. This may increase the smoothness chance for polynomial values. We describe some methods [10, 14] to estimate and compare the quality of polynomials.

2.1. **Quality of polynomials.** The quality of the chosen polynomials in polynomial selection can be modelled in terms of size and root properties [14].

2.1.1. *Size property.* Let (a, b) be pairs of relatively prime integers in the sieving region Ω. For the moment, we assume that a rectangle sieving region is used where $|a| \leq U$ and $0 < b \leq U$. 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

$$\frac{6}{\pi^2} \iint\limits_{\Omega} \rho\left(\frac{\log|F(x,y)|}{\log B_1}\right) \rho\left(\frac{\log|G(x,y)|}{\log B_2}\right) \mathrm{d}x\, \mathrm{d}y. \tag{2.1}$$

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

$$\iint\limits_{\Omega} \rho\left(\frac{\log|F(x,y)|}{\log B_1}\right) \mathrm{d}x\, \mathrm{d}y. \tag{2.2}$$

The base-(m_1, m_2) expansion [10, 11] can yield polynomials whose coefficients are $O(n^{1/(d+1)})$. The leading coefficients c_d and c_{d-1} are usually 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, while keeping the area of the sieving region $2U^2$. The sieving bounds become $|a| \leq U\sqrt{s}$ and $0 < b \leq U/\sqrt{s}$. Each monomial in the polynomial $F(a, b)$ is bounded by $|c_i|U^d s^{i-d/2}$.

In the integral (2.2), computing ρ is time-consuming, 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 by the logarithmic average of polynomial values across the sieving region.

$$\log\left(\iint\limits_{\Omega} |F(x,y)|\, \mathrm{d}x\, \mathrm{d}y\right).$$

For computational convenience, one can use the logarithmic L^2 norm for polynomial $F(x, y)$ by

$$\frac{1}{2}\log\left(\iint\limits_{\Omega} F^2(x,y)\, \mathrm{d}x\, \mathrm{d}y\right). \tag{2.3}$$

The logarithmic L^2-norm is influenced by the skewness and the location of real roots. The integral in (2.3) can be expressed as a polynomial in the coefficients of $F(x, y)$.

One can also change the range and shape of the integral region (the domain Ω), while keeping the skewness. We consider a modified logarithmic L^2-norm defined by

$$\frac{1}{2}\log\left(s^{-d}\int_{-1}^{1}\int_{-1}^{1} F^2(xs, y)\, \mathrm{d}x\, \mathrm{d}y\right). \tag{2.4}$$

where s is the skewness of sieving region.

The logarithmic L^2-norm given in Equation (2.4) is defined on a square domain. One can also use a variant with elliptic domain. We change to polar coordinates where $x = r\cos\theta$ and $y = r\sin\theta$.

$$\frac{1}{2}\log\left(s^{-d}\int_0^{2\pi}\int_0^1 F^2(s\cos\theta, \sin\theta)\, r^{2d+1}\,\mathrm{d}r\,\mathrm{d}\theta\right). \tag{2.5}$$

The logarithmic L^2-norm in Equation (2.4) is not exactly the same as the logarithmic L^2-norm in Equation (2.5), because the integrals are over different domains (ellipse and rectangle). They are both (but slightly different) approximations to the size of polynomials.

For a given norm defined in Equations (2.4) or (2.5), one should not only be able to estimate accurately that norm for a given skewness, but find the optimal skewness that gives the minimal norm.

For sextic polynomial, the logarithmic L^2-norm in Equation (2.5) can be expressed as

$$\begin{aligned}\frac{1}{2}\log\Big(\frac{\pi}{7168}\big(&231\,\tilde{c}_0^2 + 42\,\tilde{c}_0\tilde{c}_2 + 14\,\tilde{c}_0\tilde{c}_4 + 10\,\tilde{c}_0\tilde{c}_6 + 21\,\tilde{c}_1^2 + 14\,\tilde{c}_1\tilde{c}_3\\ &+ 10\,\tilde{c}_1\tilde{c}_5 + 7\,\tilde{c}_2^2 + 10\,\tilde{c}_2\tilde{c}_4 + 14\,\tilde{c}_2\tilde{c}_6 + 5\,\tilde{c}_3^2 + 14\,\tilde{c}_3\tilde{c}_5\\ &+ 7\,\tilde{c}_4^2 + 42\,\tilde{c}_4\tilde{c}_6 + 21\,\tilde{c}_5^2 + 231\,\tilde{c}_6^2\big)\Big)\end{aligned} \tag{2.6}$$

where $\tilde{c_i} = c_i s^{i-d/2}$.

2.1.2. *Root property.* If a polynomial $f(x)$ has many roots modulo small prime powers, the polynomial values may behave more smooth than random integers of about the same size. Boender, Brent, Montgomery and Murphy [4, 13, 14, 15] described some quantitative measures of this effect (root property).

Let p be a prime and $x \geq 0$ be an integer. We denote $cont_p(x)$ the exponent of the largest power of p dividing x and $cont_p(0) = \infty$. Let S be a set of uniformly distributed random integers. We denote $cont_p(S)$ the average p-valuation over elements of S. For a fixed prime p, the expected p-valuation $cont_p(S)$ is

$$1\cdot\left(\frac{1}{p} - \frac{1}{p^2}\right) + 2\cdot\left(\frac{1}{p^2} - \frac{1}{p^3}\right) + \cdots = \frac{1}{p-1}.$$

In the number field sieve, we want to know the expected p-valuation of homogeneous polynomial values. Let $F(x,y)$ be an algebraic polynomial and $f(x)$ be its dehomogenized polynomial. We discuss the roots of $F(x,y)$. Let $p^k \mid F(a,b)$ for some coprime integers a,b and some integer k. Then there are two cases: either $p \nmid b$ and $f(a/b) \equiv 0 \pmod{p^k}$ or $p \mid b$ and $h(b/a) \equiv 0 \pmod{p^k}$ where $h(x) = x^d f(1/x)$.

In the first case, pairs (a,b) can be identified by $(a/b \pmod{p^k}, 1)$. They are referred to as the *affine* roots. $F(x,y) \pmod{p^k}$ can have p^k possible affine roots, each of which relates to $p^k - p^{k-1}$ equivalent (a,b) pairs.

For the second case, pairs (a,b) can be identified by $\big(1, b/a \pmod{p^k}\big)$. We call them the *projective* roots. There are at most p^{k-1} projective roots. Each relates to $p^k - p^{k-1}$ equivalent (a,b) pairs.

Let $n_{p,k}$ be the number of affine and projective roots (counting without multiplicities) of $F \pmod{p^k}$ for $k \geq 1$. The expected p-valuation of homogeneous polynomial values is

$$cont_p(F) = \frac{1}{p+1}\sum_{k=1}^{\infty}\frac{n_{p,k}}{p^{k-1}}. \tag{2.7}$$

Murphy [14] defines the $\alpha(F)$ function to compare the cumulative expected p-valuation of polynomial values to random integers of similar size.

$$\alpha(F) = \sum_{\substack{p \leq P\\ p\ \text{prime}}}\left(\frac{1}{p-1} - cont_p(F)\right)\log p$$

where P is some bound. The $\alpha(F)$ score of a polynomial can be considered as the logarithmic benefit compared to using random integers

In the number field sieve, we want $\alpha(F)$ negative and large in absolute value. Barbulescu and Lachand [3] prove that F yields a high proportion of smooth values when $\alpha(F)$ is small.

2.1.3. *Combined score.* The logarithmic L^2-norm in Equation (2.4) or (2.5) can be modified to take the root property into account. Since the $\alpha(F)$ function affects the polynomial size on logarithmic scale, the combined score can be defined by adding $\alpha(F)$ to the logarithmic L^2-norm (e.g., the polynomial value $F(x,y)$ behaves like a random integer of size about $F(x,y)\,e^{\alpha(F)}$).

The combined score is only a rough estimate to compare polynomials. In practice, it is only trustful when the differences between polynomials are large. We use the combined score in Subsection 3.3 to show that the order of the size and root optimization is important.

2.1.4. *Murphy's* E *score.* Murphy's E score [14] is a (relatively) reliable ranking function to identify the best polynomials without test sieving. Taking the root property into account, one can refine the estimate in Equation (2.1) (for the number of sieving reports) by

$$\frac{6}{\pi^2}\int_{\Omega}\rho\left(\frac{\log|F(x,y)|+\alpha(F)}{\log B_1}\right)\rho\left(\frac{\log|G(x,y)|+\alpha(G)}{\log B_2}\right)\mathrm{d}x\,\mathrm{d}y. \tag{2.8}$$

For comparison, it is sufficient to drop the constant multiplier $6/\pi^2$. To approximate the integral, Murphy used a summation over a set of K sample points (x_i, y_i) where

$$\mathrm{E}(F,G)=\sum_{i=1}^{K}\rho\left(\frac{\log|F(x_i,y_i)|+\alpha(F)}{\log B_1}\right)\rho\left(\frac{\log|G(x_i,y_i)|+\alpha(G)}{\log B_2}\right).$$

Over an elliptic region, we can sample $x_i = \Omega_A\cos\theta_i$ and $y_i = \Omega_B\cos\theta_i$ where the Ω_A and Ω_B are the major and minor axes of the region. The angles θ_i sample the points on (the boundary of) the elliptic region. The Dickman-de Bruijn function $\rho(x)$ does not admit a closed form solution. An asymptotic expansion can be used to approximate its values. Murphy's E score is a better ranking function and we use it in Subsection 3.3 and Subsection 3.4 to compare polynomials.

2.2. **Optimizing the quality of polynomials.** Polynomial selection can be divided into three steps: polynomial generation, size optimization and root optimization. In polynomial generation, we generate many raw polynomials whose size is admissible. We further reduce the size of the raw polynomials in size optimization. Many polynomials can have comparable size after size optimization. We produce and choose the best polynomials in terms of root properties in root optimization.

Translation and rotation are useful to optimize the size and root properties. Let $f(x)=\sum_{i=0}^{d}c_ix^i$ and $g(x)=m_2x-m_1$ where $m_1/m_2 \pmod n$ is the common root.

Translation of $f(x)$ by k gives a new polynomial $f_k(x)$ defined by $f_k(x)=f(x+k)$. The linear polynomial $g_k(x)$ is $m_2x-m_1+km_2$. The common root becomes $m_1/m_2-k \pmod n$. Translation does not alter the root properties.

Rotation by a polynomial $\lambda(x)$ gives a new polynomial $f_{\lambda(x)}(x)$ defined by $f_{\lambda(x)}(x)=f(x)+\lambda(x)\,g(x)$. The linear polynomial is unchanged $g_{\lambda(x)}(x)=g(x)=m_2x-m_1$. The common root is unchanged. $\lambda(x)$ is often a linear or quadratic polynomial, depending on n and on the skewness of $f(x)$. Rotation can affect both size and root properties.

3. Size optimization

Polynomial generation (e.g., using Kleinjung's methods [10, 11]) gives many raw polynomials with small leading coefficients. The raw polynomials have very small $|c_d|, |c_{d-1}|$ and small $|c_{d-2}|$. The coefficients $|c_{d-3}|, \cdots, |c_0|$ are comparable to $(n/c_d)^{1/d}$. In size optimization, we want to produce polynomials with smaller logarithmic L^2-norm (e.g., Equation (2.5)) by changing the skewness, translating and rotating.

Assuming no cancellation occurs, we can approximate $|F(a,b)|\approx\sum_{i=0}^{d}|c_ia^ib^{d-i}|$; this approximation is maximal at the corner of the sieve region $a=U\sqrt{s}$, $b=U/\sqrt{s}$, where $|F(a,b)|\approx U^d\sum_{i=0}^{d}|c_is^{i-d/2}|$. For quintic polynomials, coefficients $|c_5|, |c_4|$ and $|c_3|$ are small. The next non-controlled coefficient is c_2. As $s\geq 1$, the dominant term is $|c_2|s^{-1/2}U^5$, so the contribution of c_2 on the polynomial value is already reduced by a factor of $s^{-1/2}$.

For sextic polynomials, the approximate polynomial values are dominated by the term $|c_3|U^6$ in the regions where no cancellation occurs. Here, c_3 is not controlled in the polynomial generation step, and we do not get a reduction in size like the $s^{-1/2}$ factor for quintic polynomials. Therefore, it is important to size-optimize sextic polynomials before trying to optimize the root properties.

In this paper, we focus on the size optimization of raw, sextic polynomials. Sextic polynomials are of main interest since they have been used in current record factorizations such as RSA-768 [19] and should be used for future records. Murphy [14] shows that the running time of the number field sieve, to factor an integer n with a degree-d polynomial, is about

$$\exp\left((1+o(1))\left(d\log d+\sqrt{(d\log d)^2+4\log(n^{1/(d+1)})\log\log(n^{1/(d+1)})}\right)\right).$$

With numerical calculations for various n and d, one can show that sextic polynomials are preferable for numbers between 220 and 360 decimal digits. The two challenge numbers RSA-896 (270 digits) and RSA-1024 (309 digits) are thus suitable for using sextic polynomials.

3.1. **Local descent method.** Let $f(x)$ be a sextic polynomial. We can use quadratic rotations since $c_3, \cdots, c_0$ have order $(n/c_6)^{1/6}$. A quadratic rotation is defined by

$$f_{u,v,w}(x) = f(x) + (ux^2 + vx + w)\,g(x) \tag{3.1}$$

for some integers u, v, w.

Murphy [14] used the classic multivariable optimization technique to optimize the L^2-norm. For sextic polynomials, there are five variables u, v, w, k, s, where k is the translation amount and s is the skewness; u, v, w, k are integers and s is real. The allowed range of these parameters is huge. For efficiency, we use a local descent method to optimize the size.

In each iteration, we attempt some translations k and rotations u, v, w, and descend into the local minimum in the direction determined by some k, u, v, w. During the procedure, we need to re-optimize the skewness of the polynomial. We describe the method in Algorithm 1.

Algorithm 1: Local descent method

```
Input  : polynomial pair $f(x) = \sum_{i=0}^{d} c_i x^i$ and $g(x) = m_2 x - m_1$;
Output: polynomial pair of smaller (or equal) $L^2$-norm;
1  $k = u = v = w = 1$;
2  repeat
3      $\tilde{f}(x) = f(x \pm k)$, $\tilde{g}(x) = g(x) \pm k m_2$;
4      if either $L^2(\tilde{f}) < L^2(f)$ then
5          $f = \tilde{f}$, $g = \tilde{g}$, $k = 2k$;
6      else
7          $k = \lceil k/2 \rceil$;
8      $\tilde{f}(x) = f(x) \pm u\,x^2\,g(x)$;
9      if either $L^2(\tilde{f}) < L^2(f)$ then
10         $f = \tilde{f}$, $u = 2u$;
11     else
12         $u = \lceil u/2 \rceil$;
13     Search similarly (e.g., lines 8-12) for linear and constant rotations;
14 until local minimum is found or loop limit is reached;
15 return $f(x), g(x)$;
```

The method seems to work for quintic polynomials, when the searching space is not too huge. However, it performs badly in practice for sextic polynomials. Many iterations get stuck at local minima without giving much reduction in size. We demonstrate this situation below.

We examine a data set consisting of 10^5 raw sextic polynomials for RSA-768. The polynomials are generated by CADO-NFS [1] and Msieve [17, 16] using Kleinjung's 2008 algorithm [11]. Figure 1 shows the normalized discrete density distribution of logarithmic L^2-norm for the raw and optimized (by the local descent) polynomials.

In particular, Table 1 shows that the raw polynomials have average logarithmic L^2-norm 80.75 (with standard deviation 1.00). The optimized (by the local descent method) polynomials have average logarithmic L^2-norm 80.46 (and standard deviation 1.02). It can be seen that only a few polynomials are optimized well by the local descent procedure. Many of them seem to descend to a local minimum rapidly and then get stuck. We discuss below some better methods to optimize such polynomials.

To overcome local minima, we could use some global optimization methods such as simulated annealing. However, they do not seem to work efficiently in our experiments, due to the huge search space and large coefficients.

Instead, we first translate the algebraic polynomial to increase the skewness. Heuristically, it moves away from the starting point and decreases the chance to get stuck in a local minimum. If the skewness of the polynomial is larger than the translation amount k, the translation does not affect the norm significantly. This can be seen from the coefficients of $f(x+k)$. A local optimization method such as descent can then be applied. One question is how to decide the translation amount. We describe some methods in Subsection 3.2.

3.2. **A better method.** We want to produce a polynomial with small L^2-norm by translation and rotation.

In the raw polynomial, c_0, c_1, c_2, c_3 have similar size and are much larger than c_4, c_5, c_6. In Equation (2.6), the $\tilde{c}_0, \tilde{c}_1, \tilde{c}_2$ are bounded by $\tilde{c}_3$. Therefore, the L^2-norm can be controlled by terms involving $\tilde{c}_3, \tilde{c}_4, \tilde{c}_6$ (since $|c_6| \approx |c_5| \ll |c_4|$). A lower bound, not depending on skewness, is dominated by the term $\tilde{c}_3^2 = c_3^2$. We demonstrate this situation for a raw polynomial A_{768} in Appendix A. It is a raw polynomial generated by Kleinjung's 2008 algorithm [11] that could be used for RSA-768.

Let $s = 3916800$ be the optimal skewness for the raw polynomial. We consider the relative weight of each term in Equation (2.6). The largest term is $5\tilde{c}_3^2 \approx 2.58 \times 10^{66}$, whereas the second largest term is $10\,\tilde{c}_2\tilde{c}_4 \approx 1.23 \times 10^{61}$. Hence, a small c_3 is a necessary condition for a small L^2-norm. The idea is to minimize c_3 by translation. Translation by k gives a polynomial in x whose coefficients are functions of k:

$$\begin{aligned} f(x+k) = c_6x^6 + (6c_6k + c_5)x^5 + \ & (15c_6k^2 + 5c_5k + c_4)x^4 \\ & + (20c_6k^3 + 10c_5k^2 + 4c_4k + c_3)x^3 + \ \cdots \end{aligned}$$

Let $c_i(k)$ be the coefficients of the i-th term in the translated polynomial. $c_3(k)$ of $f(x+k)$ is a cubic polynomial in k. The coefficients $c_0(k), c_1(k), c_2(k)$ will increase due to translation. We can use rotation to reduce them, if needed.

3.2.1. *Minimizing* $c_3(k)$. The cubic polynomial $c_3(k)$ has either one or three real roots. For each real root r, we choose K to be either $\lceil r \rceil$ or $\lfloor r \rfloor$, whichever minimizes $|c_3(k)|$. We translate $f(x)$ by K. The optimization is expected to work for all sextic polynomials since there exists at least one real root for a cubic polynomial.

In the cubic polynomial $c_3(k)$, the constant term c_3 is $O(m_1)$ (see Lemma 2.1 of [10]). The real root r is about $O((m_1/c_6)^{1/3})$. Hence $c_5(K)$ is bounded by $O(m_1^{1/3}c_6^{2/3}+c_5)$ and $c_4(K)$ is bounded by $O(m_1^{2/3}c_6^{1/3}+c_4)$. Empirically, c_4 is comparable to $c_4(K)$ for the raw polynomials found by algorithms [10, 11]. On the other hand, $m_1 \gg |c_6|$ and $|c_6| \approx |c_5|$ and hence the coefficient $c_5(K)$ can be much larger than c_5.

After translation, $c_3(k)$ is minimized and often smaller than the original c_3. Let $\delta = K - r$ and hence $|\delta| < 1$. It follows that

$$\begin{aligned} |c_3(K)| &= |20c_6K^3 + 10c_5K^2 + 4c_4K + c_3| \\ &= |20c_6(\delta^3 + 3r^2\delta + 3r\delta^2) + 10c_5(2r\delta + \delta^2) + 4c_4\delta| \\ &\leq 20\,|c_6|\,(1 + 3r^2 + 3\,|r|) + 10\,|c_5|\,(2\,|r| + 1) + 4\,|c_4|. \end{aligned} \tag{3.2}$$

Given $r = O((m_1/c_6)^{1/3})$ where $m_1 \gg |c_6|$ and $|c_5| \approx |c_6|$, Equation (3.2) above has order $O(m_1^{2/3}c_6^{1/3} + c_4)$. $|c_3(K)|$ is likely to be smaller than $|c_3| = O(m_1)$ since $|c_4| < |c_3|$ in the raw polynomial. Assume further that $|c_4| = O(m_1^{2/3}c_6^{1/3})$, which appears to be practical (see Kleinjung's 2008 method [11]). After translation, $|c_3|$ can be reduced by a factor of $(m_1/c_6)^{1/3}$.

Once K is fixed in minimizing $c_3(k)$, we can further optimize the polynomial locally by the local descent method. In the translated polynomial $f_K(x)$, the coefficients $c_5(K) = O(m_1^{1/3}c_6^{2/3})$, $c_4(K) = O(m_1^{2/3}c_6^{1/3})$, $c_3(K) = O(m_1^{2/3}c_6^{1/3})$, $c_2(K) = O(m_1^{4/3}/\,c_6^{1/3})$, $c_1(K) = O(m_1^{5/3}/\,c_6^{2/3})$, $c_0(K) = O(m_1^2/c_6)$. The coefficients $c_2(K), c_1(K), c_0(K)$ are increased during the translation. We can reduce them using rotation in the local optimization. We apply a quadratic rotation on $f_K(x)$ to reduce $c_0(K), c_1(K), c_2(K)$ to $O(m_1)$. The quadratic polynomial $ux^2 + vx + w$ used in the rotation has parameters $w = O(m_1/c_6)$, $v = O((m_1/c_6)^{2/3})$ and $u = O((m_1/c_6)^{1/3})$ (using $m_2 \ll m_1$). Let the rotated polynomial be $\tilde{f}_K(x)$ whose coefficients are $\tilde{c}_i(K)$ for $0 \leq i \leq 6$. The coefficient $\tilde{c}_3(K) = O(m_1^{2/3} c_6^{1/3} + m_2(m_1/c_6)^{1/3})$ is comparable to $c_3(K)$ if $m_2 \lesssim m_1^{1/3} c_6^{2/3}$ (parameters m_2 and c_6 can be chosen in Kleinjung's 2008 method [11]). Hence $c_0(K), c_1(K), c_2(K)$ are reduced to $O(m_1)$ without increasing too much $c_3(K)$. Comparing $\tilde{f}_K(x)$ to the raw polynomial $f(x)$, the coefficient $\tilde{c}_5(K)$ is increased, while $\tilde{c}_3(K)$ is often smaller. If the gain from a smaller $\tilde{c}_3(K)$ exceeds the deterioration from a larger $\tilde{c}_5(K)$, the L^2-norm can be reduced. In practice, the local descent method (Algorithm 1) can be applied, instead of applying a single rotation.

We give an example for the polynomial A_{768} in Appendix A. It has logarithmic L^2-norm 72.59. The coefficient of x^3 in $f(x+k)$ is

$$\begin{aligned} &71727600k^3 + 190647000k^2 + 112950493882223418033937 2k+ \\ &718693701130240225274612814188142. \end{aligned}$$

The cubic polynomial has a real root near $k = -191352410$. We translate $f(x)$ by k and then apply the local descent method to the translated polynomial. The resulting optimized polynomial B_{768} in Appendix A has logarithmic L^2-norm 67.60.

The method works better on average than the local descent method used alone. Table 1 (also c.f. Figure 1) shows that the improved method can reduce the average logarithmic L^2-norm to 70.34 (of standard deviation 0.60). We gain almost 10 on the average logarithmic L^2-norm compared to the local descent method used alone.

3.2.2. *Further improvement.* Thorsten Kleinjung describes a method (personal communication) which helps further reduce the norm. Before translation, we attempt several cubic rotations by $f(x) + \delta x^3 g(x)$ for small δ's on the raw polynomial $f(x)$. This gives some variation for the size optimization. For each rotated polynomial, we repeat the optimization procedure from Subsection 3.2 and record the minimum norm found. The variation gives some benefits in practice. We consider the same data set of 10^5 polynomials used previously. In experiments, we rotate polynomials by $|\delta| \leq 256$ and optimize the size using the above method.

Table 1. Comparison of methods for size optimization. Column $\log(L^2)$ records the average logarithmic L^2-norm of polynomials after size optimization; Column C and E are the average combined score (c.f. Subsubsection 2.1.3) and Murphy's E score (c.f. Subsubsection 2.1.4) for polynomials after size optimization. Here we use $B_1 = 1.1 \times 10^9$, $B_2 = 2.0 \times 10^8$ and the area 2.362×10^{18} for the domain Ω in the computation of Murphy's E score.

	$\log(L^2)$	$\alpha(F)$	C	E
Raw polynomials	80.75	-0.513	80.24	8.35e-15
Method of Subsection 3.1	80.46	-0.510	79.96	7.06e-15
Method of Subsubsection 3.2.1	70.34	-0.512	69.82	3.51e-14
Method of Subsubsection 3.2.2	69.84	-0.516	69.33	3.33e-14

Table 1 and Figure 1 show this variant further reduces the average logarithmic L^2-norm to 69.84 (of standard deviation 0.56). We gain another 0.5 on the average logarithmic L^2-norm compared to the above method.

3.3. **Order of optimization.** The raw polynomial often has a small c_5, which permits a larger rotation space in root optimization. The size optimization procedure leads to a much larger c_5 due to translation. This may lead to a smaller rotation space. Therefore, it is suspected that we could have optimized the root property (of the raw polynomial) first and then optimized the size (by using only translation and changing the skewness). If the root property is outstanding, changing the order of optimization may yield better results. However, we give some heuristic argument to show that this is unlikely in practice.

Let $f_{u,v,w}(x)$ be the rotated polynomial in Equation (3.1). If $c_3 = O(m_1)$ and $c_0 \approx c_3 s^3$, $c_1 \approx c_3 s^2$, $c_2 \approx c_3 s$, we have an upper bound $O(s^6)$ for the rotation space. We need to estimate the expected minimum $\alpha(F)$ after $K \approx s^6$ random polynomials are chosen.

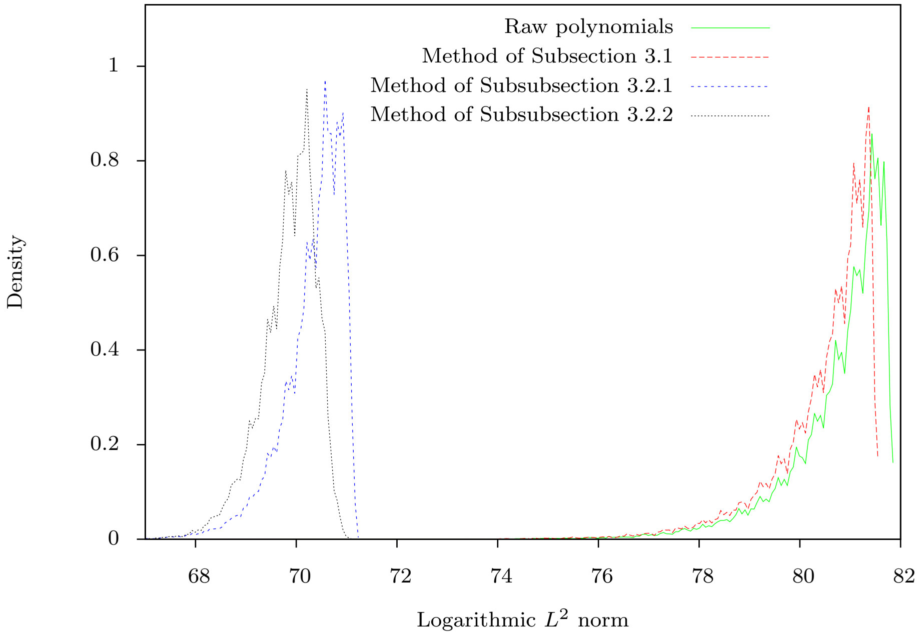

FIGURE 1. Distribution of logarithmic L^2-norm after size optimization

Emmanuel Thomé described a method (personal communication) to estimate the expected minimum of $\alpha(F)$ using order statistics. We describe his method here.

We assume that the $\alpha(F)$ values of random polynomials follow a standard Gaussian (normal) distribution $N(\mu, \sigma^2)$ (see Figure 2). Let $\Phi(x)$ be the cumulative density function for the standard $N(0,1)$ normal distribution $\phi(x)$ where

$$\phi(x) = \frac{1}{\sqrt{2\pi}}\, e^{-x^2/2}, \quad \Phi(x) = \frac{1}{2}\left(1 + \operatorname{erf}\left(\frac{x}{\sqrt{2}}\right)\right).$$

In function $\Phi(x)$, $\operatorname{erf}(x)$ is the error function [7] defined by

$$\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2}\, \mathrm{d}t.$$

The probability distribution for the minimum order statistic is given by

$$p_K(x) = K\, \left(1 - \Phi(x)\right)^{K-1}\, \phi(x)$$

where K is the cardinality of the sample set. We use an asymptotic approximation [6] for the expected value of the minimum order statistic of the normal distribution:

$$\mu - \sigma \left(\sqrt{2 \log K} - \frac{\log(\log K) + 1.377}{2\sqrt{2 \log K}} \right). \tag{3.3}$$

In practice, we need to estimate the actual parameters μ, σ for the distribution. We consider the 10^5 raw polynomials for RSA-768 used in Subsection 3.1. Figure 2 shows that empirically the distribution of $\alpha(F)$ is close to a Gaussian distribution. The polynomials have mean $\mu = -0.513$ and standard deviation $\sigma = 0.816$. Here the average $\alpha(F)$ is negative since the raw polynomials are generated in a way such that they are expected to have good projective root property (e.g., c_d is divisible by many small primes).

We use these parameters ($\mu = -0.513$ and $\sigma = 0.816$) to estimate the expected value of the minimum order statistic. Equation (3.3) shows that the expected minimum α is proportional to the square root of logarithmic scale of skewness.

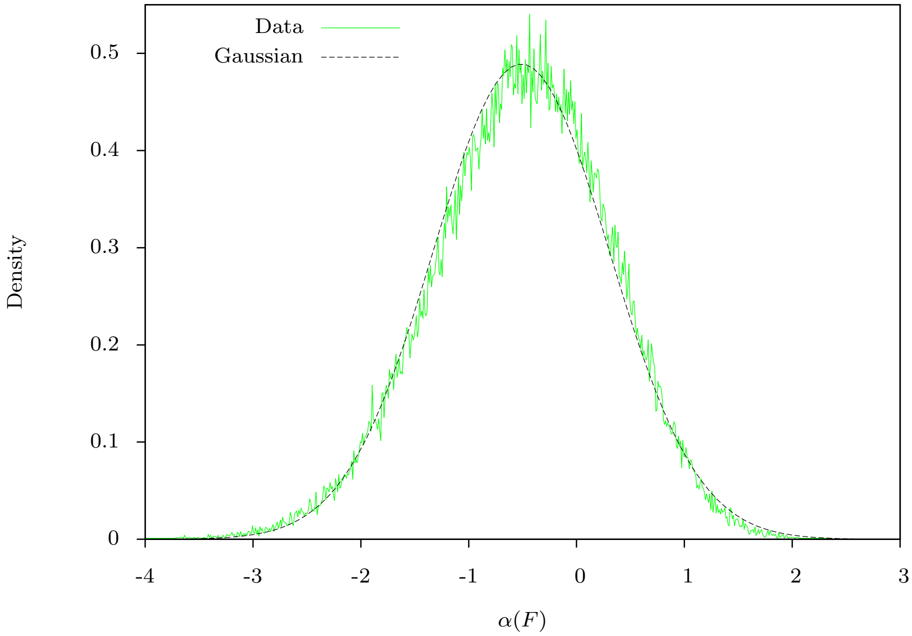

FIGURE 2. Distribution of α of 10^5 raw RSA-768 polynomials

For raw polynomials, the logarithmic L^2-norm (c.f. Equation (2.5)) is bounded by $\log m_1$. After size optimization, the bound can be reduced to $\log(m_1^{2/3}c_6^{1/3})$ if $m_2 \leq m_1^{1/3}c_6^{2/3}$. The skewness can be bounded by $(m_1/c_6)^{2/9}$ and hence the expected minimum α is bounded by $-\sigma\sqrt{\frac{8}{3}\log(m_1/c_6)}$.

If we first conduct root optimization, the skewness $(m_1/c_6)^{1/3}$ gives expected minimum α of about $-\sigma\sqrt{4\log(m_1/c_6)}$. This is only slightly better than the above case. Then we can apply a size optimization of two variables (translation and skewness) afterwards. However, such optimization is restricted as no rotation can be used any more (since we do not want to change the good root properties).

We also confirm above heuristic argument by experiments. For the RSA-768 data, we optimize the polynomials in two ways: size-root optimization and root-size optimization respectively. In the raw polynomials, the average $c_6 \approx 1.66 \times 10^8$ and average $m_1 \approx 1.41 \times 10^{37}$.

TABLE 2. Size-root and root-size optimization. The rows "sopt" and "ropt" denotes size and root optimization respectively; The rows "Size-root" and "Root-size" represents the order of optimization. Columns $\log(L^2)$ and $\alpha(F)$ record the average logarithmic L^2-norm and α-score of polynomials after size-root optimization; Columns C and E are the average combined score (c.f. Subsubsection 2.1.3) and Murphy's E score (c.f. Subsubsection 2.1.4) for polynomials after size-root optimization; Columns Top C and Top E are the average combined score and Murphy's E score for the top 100 polynomials respectively.

		$\log(L^2)$	Skew	$\alpha(F)$	C	Top C	E	Top E
Raw polynomials (c.f Table 1)		80.75	8.35e+5	-0.513	80.24	73.04	8.35e-15	3.46e-14
Size-root	(1) sopt (c.f Table 1)	69.84	1.80e+4	-0.516	69.33	65.51	3.33e-14	7.08e-14
	(2) ropt (c.f Table 3)	71.86	7.43e+4	-7.019	64.84	60.57	8.60e-14	2.14e-13
Root-size	(1) ropt	80.78	5.95e+7	-8.050	72.73	64.82	4.91e-14	1.57e-13
	(2) sopt	80.70	1.15e+7	-8.050	72.65	64.61	3.32e-14	1.46e-13

Table 2 (and also Figure 1) shows that a reduction of about 11 in the logarithmic norm is achievable in size optimization. A following root optimization can further reduce the combined score (c.f. Subsubsection 2.1.3), despite slightly increasing the size. If we first optimize root, the (average) $\alpha(F)$ achieved can be slightly better due to a larger rotation space. However, the following size optimization is not effective (average logarithmic norm reduced from 80.78 to 80.70) since we can only use translation then. It is also noted that the Murphy's E scores (in the last row of Table 2) become worse since we optimize the norm only during size optimization.

Put together, size-root (in order) optimization behaves much better than root-size optimization in practice. Therefore, it is suggested to optimize the size property first and then the root property.

3.4. **Comparison.** We further consider some experiments to compare the final polynomials (after size and root optimization) obtained by the new method (c.f. Subsubsection 3.2.2) and the local descent method. For the RSA-768 data, we optimize the size of the raw polynomials in both ways and then run the root optimization on the size-optimized polynomials. The order of the optimization will be justified in Subsection 3.3. The scores of the polynomials after root optimization are tabulated in Table 3. Assuming that Murphy's E score provides an accurate estimation of the yield rate. We can see that the new method produces polynomials with higher yield rates on average. It is expected that the we can find a better polynomial.

Table 3. Comparison of two optimization methods. The columns use a similar notation as Table 2.

	$\log(L^2)$	$\alpha(F)$	C	Top C	E	Top E
Method of Subsection 3.1	80.65	-7.777	72.87	64.23	3.37e-14	1.53e-13
Method of Subsubsection 3.2.2 (c.f. Table 2)	71.86	-7.019	64.84	60.57	8.60e-14	2.14e-13

4. Conclusion

We described some better methods to optimize the size by determining an appropriate initial polynomial for the iteration and then locally optimizing the polynomial. The method described in Subsection 3.2 is implemented in CADO-NFS [1], an open-source implementation of the number field sieve, which has been used in the factorization of some large integers. For instance, the factorization [2] of the 704-bit RSA challenge number in July 2012 used this method to optimize the size of raw polynomials.

Acknowledgements

The authors are grateful to Richard Brent, Steven Galbraith, Thorsten Kleinjung, Jason Papadopoulos and Emmanuel Thomé for helpful comments and discussions, and to Jean-Charles Gilbert for giving pointers to Lasserre's relaxation. The first author would like to thank the Australian National University, NeSI (New Zealand eScience Infrastructure) and the Centre for eResearch at the University of Auckland for providing funding and computing facilities.

Appendix A. Polynomials

The appendix contains polynomial pairs A_{768}, B_{768} discussed in the paper.

Polynomial A_{768}:

$$\begin{aligned}
f(x) &= 3586380\,x^6 + 19064700x^5 + 282376234705558545084843x^4 \\
&+ 718693701130240225274612814188142x^3 \\
&+ 4340200162893339761259991222380911282x^2 \\
&- 12541568233611627968693736065307030120x \\
&+ 9008374174467563445936947139641332877 \\
g(x) &= 533620548325820192253 83\,x \\
&- 26457722251514149087911384249044520830 \\
\text{skewness: } & 3916800.00 \\
L^2\text{-norm: } & 72.59 \\
\alpha(F)\text{: } & -1.08
\end{aligned}$$

Polynomial B_{768}:

$$\begin{aligned}
f(x) &= 3586380\,x^6 - 4117247962908300x^5 + 2251833225235534190109843x^4 \\
&\quad + 136220930040469670784138610516x^3 \\
&\quad - 1750146668953172123277757169064100 7037x^2 \\
&\quad - 2611070303825589994778764286883040274767 31x \\
&\quad + 761551516028003977492805501977631103604836 3657612 \\
g(x) &= 53362054832582019225383\,x \\
&\quad - 264577324616616410519945277304773 03005 \\
\text{skewness: } & 2593792.00 \\
L^2\text{-norm: } & 67.60 \\
\alpha(F)\text{: } & -2.04
\end{aligned}$$

References

1. 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. Release 2.0, available from `http://cado-nfs.gforge.inria.fr`, 2013.
2. S. Bai, E. Thomé, P. Zimmermann. Factorisation of RSA-704 with CADO-NFS. Report, 2012. `http://eprint.iacr.org/2012/369.pdf.`
3. R. Barbulescu and A. Lachand. Some mathematical remarks on the polynomial selection in NFS. Report, 2014. `http://arxiv.org/abs/1403.0184.`
4. H. Boender. *Factoring large integers with the quadratic sieve*. PhD thesis, Leiden University, 1997.
5. J. Buhler, H. Lenstra, and C. Pomerance. Factoring integers with the number field sieve. In Lenstra and Lenstra [12], pages 50–94.
6. H. Cramér. *Mathematical Methods of Statistics*. Princeton University Press, 1999.
7. W. Gautschi. Chapter 7. Error function and Fresnel integrals. In M. Abramowitz and I. A. Stegun, editors, *Handbook of Mathematical Functions: with Formulas, Graphs, and Mathematical Tables*, page 1046. Dover Publications, 1972.
8. A. Granville. Smooth numbers: computational number theory and beyond. In *Proc. MSRI Conf. Algorithmic Number Theory: Lattices, Number Fields, Curves and Cryptography*. MSRI Publications, Volume 44, 2008.
9. A. Hildebrand and G. Tenenbaum. Integers without large prime factors. *Journal de Théorie des Nombres de Bordeaux*, 5(2):411–484, 1993.
10. T. Kleinjung. On polynomial selection for the general number field sieve. *Mathematics of Computation*, 75(256):2037–2047, 2006.
11. T. Kleinjung. Polynomial selection. In *CADO workshop on integer factorization*, INRIA Nancy, 2008. `http://cado.gforge.inria.fr/workshop/slides/kleinjung.pdf`.
12. A. K. Lenstra and H. W. Lenstra, Jr., editors. *The Development of the Number Field Sieve*, volume 1554 of *Lecture Notes in Mathematics*. Springer, 1993.
13. B. A. Murphy. Modelling the Yield of Number Field Sieve Polynomials. In *Algorithmic Number Theory - ANTS III, LNCS 1443*, pages 137–147, 1998.
14. B. A. Murphy. *Polynomial selection for the number field sieve integer factorisation algorithm*. PhD thesis, The Australian National University, 1999.
15. B. A. Murphy and R. P. Brent. On quadratic polynomials for the number field sieve. In *Proceedings of the CATS '98*, volume 20 of *Australian Computer Science Communications*, pages 199–213. Springer, 1998.
16. J. Papadopoulos. Call for volunteers: RSA768 polynomial selection, 2011. `http://www.mersenneforum.org/showthread.php?t=15540`.
17. J. Papadopoulos. Msieve v1.48, 2011. `http://sourceforge.net/projects/msieve`.
18. J. M. Pollard. The lattice sieve. In Lenstra and Lenstra [12], pages 43–49.
19. 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. J. J. te Riele, A. Timofeev, and P. Zimmermann. Factorization of a 768-bit RSA modulus. In *Proceedings of CRYPTO '10*, volume 6223 of *Lecture Notes in Computer Science*, pages 333–350. Springer, 2010.

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

INRIA Nancy - Grand Est, Villers-les-Nancy, France.
E-mail address: cyril.bouvier@inria.fr

INRIA Nancy - Grand Est, Villers-les-Nancy, France.
E-mail address: alexander.kruppa@inria.fr

INRIA Nancy - Grand Est, Villers-les-Nancy, France.
E-mail address: paul.zimmermann@inria.fr