New Complexity Trade-Offs for the (Multiple) Number Field Sieve Algorithm in Non-Prime Fields

Palash Sarkar and Shashank Singh

Applied Statistics Unit
Indian Statistical Institute
palash@isical.ac.in, sha2nk.singh@gmail.com

Abstract. The selection of polynomials to represent number fields crucially determines the efficiency of the Number Field Sieve (NFS) algorithm for solving the discrete log problem in a finite field. An important recent work due to Barbulescu et al builds upon existing works to propose two new methods for polynomial selection when the target field has a composite order. These methods are called the generalised Joux-Lercier (GJL) and the Conjugation methods. In this work, we propose a new method for polynomial selection for the NFS algorithm in fields $\mathbb{F}_{p^n}$, with $n > 1$. The new method both subsumes and generalises the GJL and the Conjugation methods. Asymptotic analysis for the new polynomial selection method is performed for both the classical NFS and the multiple NFS (MNFS) algorithms. For medium and large prime characteristic, the new method does not provide any new asymptotic result. For the boundary case, the complexity of the new method interpolates between that of the Conjugation and the GJL methods for both classical and multiple NFS algorithms. In particular, for $p = L_Q(2/3, c_p)$, as c_p grows the complexity of the new method is lower than that of the GJL method and hence becomes the new state of the art.

1 Introduction

Let $G = \langle g \rangle$ be a finite cyclic group. The discrete log problem (DLP) in G is the following. Given (g, h), compute the minimum non-negative integer a such that $h = g^a$. For appropriately chosen groups G, the DLP in G is believed to be computationally hard. This forms the basis of security of many important cryptographic protocols.

Studying the hardness of the DLP on subgroups of the multiplicative group of a finite field is an important problem. There are two general algorithms for tackling the DLP on such groups. These are the function field sieve (FFS) [1, 2, 12, 15] algorithm and the number field sieve (NFS) [7, 13, 17] algorithm. Both these algorithms follow the framework of index calculus algorithms which is currently the standard approach for attacking the DLP in various groups.

For small characteristic fields, the FFS algorithm leads to a quasi-polynomial running time [5]. Using the FFS algorithm outlined in [11, 5], Granger et al [8]

reported a record computation of discrete log in the binary extension field $\mathbb{F}_{2^{9234}}$. FFS also applies to the medium characteristic fields. Some relevant works on this line are reported in [15, 10, 22].

For medium to large characteristic finite fields, the NFS algorithm is the state-of-the-art. In the context of the DLP, the NFS was first proposed by Gordon [7] for prime order fields. The algorithm proceeded via number fields and one of the main difficulties in applying the NFS was in the handling of units in the corresponding ring of algebraic integers. Schirokauer [23, 25] proposed a method to bypass the problems caused by units. Further, Schirokauer [24] showed the application of the NFS algorithm to composite order fields. Joux and Lercier [13] presented important improvements to the NFS algorithm as applicable to prime order fields.

Joux, Lercier, Smart and Vercauteren [17] later showed that the NFS algorithm is applicable to all finite fields. Since then, several works [18, 4, 9, 21] have gradually improved the NFS in the context of medium to large characteristic finite fields.

The efficiency of the NFS algorithm is crucially dependent on the properties of the polynomials used to construct the number fields. Consequently, polynomial selection is an important step in the NFS algorithm and is an active area of research. The recent work [4] extends a previous method [13] for polynomial selection and also presents a new method. The extension of [13] is called the generalised Joux-Lercier (GJL) method while the new method proposed in [4] is called the conjugation method. The paper also provides a comprehensive comparison of the trade-offs in the complexity of the NFS algorithm offered by the various polynomial selection methods.

The NFS based algorithm has been extended to multiple number field sieve algorithm (MNFS). The work [6] showed the application of the MNFS to medium to high characteristic finite fields. More recently, Pierrot [21] proposed MNFS variants of the GJL and the Conjugation methods.

Our contributions: In this work, we build on the works of [13, 4] to propose a new method of polynomial selection for NFS over $\mathbb{F}_{p^n}$. The new method both subsumes and generalises the GJL and the Conjugation methods. There are two parameters to the method, namely a divisor d of the extension degree n and a parameter $r \geq k$ where $k = n/d$.

For $d = 1$, the new method becomes the same as the GJL method. For $d = n$ and $r = k = 1$, the new method becomes the same as the Conjugation method. For $d = n$ and $r > 1$; or, for $1 < d < n$, the new method provides polynomials which leads to different trade-offs than what was previously known. Note that the case $1 < d < n$ can arise only when n is composite, though the case $d = n$ and $r > 1$ arises even when n is prime. So, the new method provides new trade-offs for both n composite and n prime.

Following the works of [4, 21] we carry out an asymptotic analysis of new method for the classical NFS as well as for MNFS. For the medium and the large characteristic cases, the results for the new method are exactly the same as those obtained in [4, 21]. For the boundary case, however, we obtain some

interesting asymptotic results. We discuss these for the MNFS though similar results also hold for the NFS.

Let $p = L_Q(2/3, c_p)$ and let θ_0 and θ_1 be such that the complexity of the MNFS-Conjugation method is $L_Q(1/3, \theta_0)$ and the complexity of the MNFS-GJL method is $L_Q(1/3, \theta_1)$. As shown in [21], $L_Q(1/3, \theta_0)$ is the minimum complexity of MNFS[1] while for large c_p, the complexity of MNFS-GJL is the smallest.

For MNFS-New with $k = 1$, let the complexity be $L_Q(1/3, C(r))$ where for a fixed r, $C(r)$ is the minimum over all c_p. Then we show that $C(r)$ is monotone increasing for $r \geq 1$; $C(1) = \theta_0$; and that $C(r)$ is bounded above by θ_1 which is its limit as r goes to infinity. So, for the new method the minimum complexity is the same as MNFS-Conjugation method. On the other hand, as r increases, the complexity of MNFS-New becomes the minimum of the complexities of all the prior known methods. In particular, the complexity of MNFS-New interpolates nicely between the complexities of the MNFS-GJL and the MNFS-Conjugation methods.

2 Background on NFS for Non-Prime Fields

We provide a brief sketch of the background on the variant of the NFS algorithm that is applicable to the extension fields $\mathbb{F}_Q$, where $Q = p^n$, p is a prime and $n > 1$. More detailed discussions can be found in [13, 4].

Following the structure of index calculus algorithms, NFS has three main phases, namely, relation collection (sieving), linear algebra and descent. Prior to these, is the set-up phase. In the set-up phase, two number fields are constructed and the sieving parameters are determined. The two number fields are set up by choosing two irreducible polynomials $f(x)$ and $g(x)$ over the integers such that their reductions modulo p have a common irreducible factor $\phi(x)$ of degree n over $\mathbb{F}_p$. The field $\mathbb{F}_{p^n}$ will be considered to be represented by $\phi(x)$.

The choices of the two polynomials $f(x)$ and $g(x)$ are crucial to the algorithm. These greatly affect the overall run time of the algorithm. Let $\alpha, \beta \in \mathbb{C}$ and $m \in \mathbb{F}_{p^n}$ be the roots of the polynomials $f(x)$, $g(x)$ and $\phi(x)$ respectively. The two number fields and the finite field are given as follows.

$$\mathbb{K}_1 = \mathbb{Q}(\alpha) = \frac{\mathbb{Q}[x]}{\langle f(x) \rangle}, \mathbb{K}_2 = \mathbb{Q}(\beta) = \frac{Q[x]}{\langle g(x) \rangle} \text{ and } \mathbb{F}_{p^n} = \mathbb{F}_p(m) = \frac{\mathbb{F}_p[x]}{\langle \phi(x) \rangle}.$$

Thus, there are two homomorphisms $\psi_1 : \mathbb{K}_1 \to \mathbb{F}_{p^n}$ and $\psi_2 : \mathbb{K}_2 \to \mathbb{F}_{p^n}$ given by $\alpha \hookrightarrow m$ and $\beta \hookrightarrow m$ respectively. Actual computations are carried out over these number fields and are then transformed to the finite field via these homomorphisms. In fact, instead of doing the computations over the whole number field $\mathbb{K}_i$, one works over its ring of algebraic integers $\mathbb{O}_i$. These integer rings provide a nice way of constructing a factor basis and moreover, unique factorisation of ideals holds over these rings.

[1] The value of θ_0 obtained in [21] is incorrect.

The factor basis $\mathcal{F} = \mathcal{F}_1 \cup \mathcal{F}_2$ is chosen as follows.

$$\mathcal{F}_1 = \{\text{prime ideal } \overline{\mathfrak{q}}_i \ \in \mathbb{O}_1 \text{ having norm less than } B\} \tag{1}$$

$$\mathcal{F}_2 = \{\text{prime ideal } \overline{\mathfrak{s}}_j \ \in \mathbb{O}_2 \text{ having norm less than } B\} \tag{2}$$

where B is the smoothness bound and is to be chosen appropriately. An algebraic integer is said to be B-smooth if the principal ideal generated by it factors into the prime ideals of norms less than B. As a result the size of the factor basis is at most $2B$. For asymptotic computations, this is simply taken to be B. The work flow of NFS can be understood by the diagram in Figure 1.

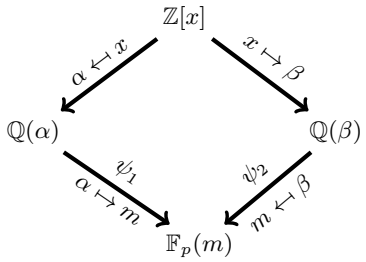

Fig. 1. A work-flow of NFS.

A polynomial $\lambda(x) \in \mathbb{Z}[x]$ of degree $t-1$ (i.e. having t coefficients) is chosen and the principal ideals generated by its images in the two number field are checked for smoothness. If both of these are smooth, then

$$\lambda(\alpha)\mathbb{O}_1 = \prod_i \overline{\mathfrak{q}}_i^{\ e_i} \text{ and } \lambda(\beta)\mathbb{O}_2 = \prod_j \overline{\mathfrak{s}}_j^{\ e'_j} \tag{3}$$

where $\overline{\mathfrak{q}}_i$ and $\overline{\mathfrak{s}}_j$ are the prime ideals of the integer rings $\mathbb{O}_1$ and $\mathbb{O}_2$ respectively.

Let h_1 and h_2 be the class numbers of the number fields $\mathbb{O}_1$ and $\mathbb{O}_2$ respectively. Then

$$\Big(\lambda(\alpha)\mathbb{O}_1\Big)^{h_1} = \Big(\prod_i \overline{\mathfrak{q}}_i^{\ e_i}\Big)^{h_1} \text{ and } \Big(\lambda(\beta)\mathbb{O}_2\Big)^{h_2} = \Big(\prod_j \overline{\mathfrak{s}}_j^{\ e'_j}\Big)^{h_2}. \tag{4}$$

This leads to

$$\lambda(\alpha)^{h_1}\mathbb{O}_1 = \prod_i \Big(\overline{\mathfrak{q}}_i^{\ h_1}\Big)^{e_i} \text{ and } \lambda(\beta)^{h_2}\mathbb{O}_2 = \prod_j \Big(\overline{\mathfrak{s}}_j^{\ h_2}\Big)^{e'_j}. \tag{5}$$

Note that $(\overline{\mathfrak{q}}_i^{\ h_1})$ and $(\overline{\mathfrak{s}}_j^{\ h_2})$ are principal ideals of $\mathbb{O}_1$ and $\mathbb{O}_2$ respectively. Let $(\overline{\mathfrak{q}}_i^{\ h_1}) = \delta_i\mathbb{O}_1$ and $(\overline{\mathfrak{s}}_j^{\ h_2}) = \epsilon_j\mathbb{O}_2$. Thus,

$$\lambda(\alpha)^{h_1}\mathbb{O}_1 = \prod_i {\delta_i}^{e_i}\mathbb{O}_1 \text{ and } \lambda(\beta)^{h_2}\mathbb{O}_2 = \prod_j \epsilon_j^{e'_j}\mathbb{O}_2. \tag{6}$$

Converting (6) into element form results in

$$\lambda(\alpha)^{h_1} = u \prod_i {\delta_i}^{e_i} \text{ and } \lambda(\beta)^{h_2} = v \prod_j \epsilon_j^{e'_j} \tag{7}$$

where u and v are units in the rings $\mathbb{O}_1$ and $\mathbb{O}_2$ respectively.

Let (r_1, r_2) and (s_1, s_2) be the signatures of $\mathbb{O}_1$ and $\mathbb{O}_2$ respectively. Then $\mathbb{O}_1^\star \equiv \mu(\mathbb{K}_1) \times \mathbb{Z}^{r_1+r_2-1}$, where $\mu(\mathbb{K}_1) = \langle u_0 \rangle$ is a finite cyclic group. Let $r = r_1 + r_2 - 1$. Assuming that we can compute a system of fundamental units $u_1, u_2, \ldots, u_r$, we can write $u = \prod_{i=0}^{r} u_i^{l_i}$. Similarly we have $\mathbb{O}_2^\star \equiv \mu(\mathbb{K}_2) \times \mathbb{Z}^{s_1+s_2-1}$. Letting $s = s_1 + s_2 - 1$, $\mu(\mathbb{K}_2) = \langle v_0 \rangle$, and $v_1, v_2, \ldots, v_s$ as a fundamental units, we can write $v = \prod_{j=0}^{s} v_j^{l'_j}$. Putting the values in (7) provides the following relations.

$$\lambda(\alpha)^{h_1} = \prod_{i=0}^{r} u_i^{l_i} \prod_i {\delta_i}^{e_i} \text{ and } \lambda(\beta)^{h_2} = \prod_{j=0}^{s} v_j^{l'_j} \prod_j \epsilon_j^{e'_j}. \tag{8}$$

Using the homomorphisms ψ_i, the relations given by (8) can be converted to the finite field $\mathbb{F}_{p^n}$ represented by the polynomial $\phi(x)$.

$$\psi_1(\lambda(\alpha))^{h_1} = \prod_{i=0}^{r} \psi_1(u_i)^{l_i} \prod_i \psi_1(\delta_i)^{e_i} \text{ and } \psi_2(\lambda(\beta))^{h_2} = \prod_{j=0}^{s} \psi_2(v_j)^{l'_j} \prod_j \psi_2(\epsilon_j)^{e'_j}$$

Taking discrete logarithms on both sides result in the following equations.

$$h_1 \log\Big(\psi_1(\lambda(\alpha))\Big) = \sum_{i=0}^{r} l_i \log\left(\psi_1(u_i)\right) + \sum_i e_i \log\left(\psi_1(\delta_i)\right) \tag{9}$$

$$h_2 \log\Big(\psi_2(\lambda(\beta))\Big) = \sum_{j=0}^{s} l'_j \log\left(\psi_2(v_j)\right) + \sum_j e'_j \log\left(\psi_2(\epsilon_j)\right). \tag{10}$$

Recall that the map $\psi_1 : \alpha \hookrightarrow m$ and $\psi_2 : \beta \hookrightarrow m$. So, $\psi_1(\lambda(\alpha)) = \lambda(m) = \psi_2(\lambda(\beta))$. Using this in (9) and (10), leads to the following relation.

$$h_1 \left[\sum_{j=0}^{s} l'_j \log\left(\psi_2(v_j)\right) + \sum_j e'_j \log\left(\psi_2(\epsilon_j)\right) \right] = h_2 \left[\sum_{i=0}^{r} l_i \log\left(\psi_1(u_i)\right) + \sum_i e_i \log\left(\psi_1(\delta_i)\right) \right] \tag{11}$$

The relation given by (11) involves only the discrete log of field elements. Many such relations are collected by sieving over suitable $\lambda(x)$. The linear algebra step solves the resulting system of linear equations using either the Lanczos or the block Wiedemann algorithms to obtain the discrete logs of the field elements present in these equations.

The above description assumes the availability of the class numbers and the fundamental system of units. This may not always be the case. In such a scenario, the Schirokauer maps [23, 25] are used to write the system of linear equations in terms of virtual logarithms [25, 16]. We skip the details of virtual logarithms and Schirokauer maps, as these details will not affect the polynomial selection problem considered in this work.

After the linear algebra phase is over, the descent phase is used to compute the discrete logs of the given elements of the field $\mathbb{F}_{p^n}$. We also skip these details. For recent work on the descent phase, we refer to [9].

3 Polynomial Selection and its Effect on NFS Complexity

It is evident from the description of NFS that the relation collection phase requires polynomials $\lambda(x) \in \mathbb{Z}[x]$ whose images in the two number fields are simultaneously smooth. For ensuring the smoothness of $\lambda(\alpha)$ and $\lambda(\beta)$, it is enough to ensure that their norms viz, $\mathrm{Res}(f, \lambda)$ and $\mathrm{Res}(g, \lambda)$ are B-smooth. We refer to [4] for further explanations.

Kalkbrener in [19, Corollary 2], gave the following upper bound for the absolute value of the norm.

$$|\mathrm{Res}(f, \lambda)| \leq \kappa\left(\deg f, \deg \lambda\right) \|f\|_\infty^{\deg \lambda} \|\lambda\|_\infty^{\deg f} \tag{12}$$

where $\kappa(a, b) = \binom{a+b}{a}\binom{a+b-1}{a}$ and $\|f\|_\infty$ is maximum of the absolute values of the coefficients of f.

Following [4], let E be such that the coefficients of λ are in $\left[-\frac{1}{2}E^{2/t}, \frac{1}{2}E^{2/t}\right]$. So, $\|\lambda\|_\infty = E^{2/t}$ and the number of polynomials $\lambda(x)$ that is considered for the sieving is E^2.

Whenever $p = L_Q(a)$ with $a > \frac{1}{3}$, we have the following bound on the $\mathrm{Res}(f, \lambda) \times \mathrm{Res}(g, \lambda)$ (for details we refer to [4]).

$$|\mathrm{Res}(f, \lambda) \times \mathrm{Res}(g, \lambda)| \approx \left(\|f\|_\infty \|g\|_\infty\right)^{t-1} E^{(\deg f + \deg g)2/t}. \tag{13}$$

For small values of n, the sieving polynomial $\lambda(x)$ is taken to be linear, i.e., $t = 2$ and then the norm bound becomes approximately $\|f\|_\infty \|g\|_\infty E^{(\deg f + \deg g)}$.

The methods for choosing f and g result in the coefficients of one or both of these polynomials to depend on Q. So, the right side of (13) is determined by Q and E. All polynomial selection algorithms try to minimize the RHS of (13). From the bound in (13), it is evident that during polynomial selection, the goal should be to try and keep the degrees and the coefficients of both f and g to be small. Ensuring both degrees and coefficients to be small is a nontrivial task and leads to a trade-off. Previous methods for polynomial selections provide different trade-offs between the degrees and the coefficients.

Estimates of Q-E trade-off values have been provided in [4] and is based on the CADO factoring software [3]. Table 1 reproduces these values.
As mentioned in [4, 9], presently the following three polynomial selection methods provide competitive trade-offs.

Table 1. Estimate of Q-E values [4].

Q(dd)	100	120	140	160	180	200	220	240	260	280	300
Q(bits)	333	399	466	532	598	665	731	798	864	931	997
E(bits)	20.9	22.7	24.3	25.8	27.2	28.5	29.7	30.9	31.9	33.0	34.0

1. **JLSV1:** Joux, Lercier, Smart, Vercauteren method [16].
2. **GJL:** Generalised Joux Lercier method [20, 4].
3. **Conjugation method** [4]

Brief descriptions of these methods are given below.

JLSV1. Repeat the following steps until f and g are obtained to be irreducible over $\mathbb{Z}$ and ϕ is irreducible over $\mathbb{F}_p$.

1. Randomly choose polynomials $f_0(x)$ and $f_1(x)$ having small coefficients with $\deg(f_1) < \deg(f_0) = n$.
2. Randomly choose an integer a to be slightly greater than $\lceil\sqrt{p}\rceil$.
3. Let (u, v) be the rational reconstruction of a in $\mathbb{F}_p$, i.e., $a \equiv u/v \bmod p$.
4. Define $f(x) = f_0(x) + af_1(x)$ and $g(x) = vf_0(x) + uf_1(x)$ and $\phi(x) = f(x) \bmod p$.

Note that $\deg(f) = \deg(g) = n$ and both $\|f\|_\infty$ and $\|g\|_\infty$ are $O\left(p^{1/2}\right) = O\left(Q^{1/(2n)}\right)$ and so (13) becomes $E^{4n/t}Q^{(t-1)/n}$ which is $E^{2n}Q^{1/n}$ for $t = 2$.

GJL. The basic Joux-Lercier method [14] works for prime fields. The generalised Joux-Lercier method extends the basic Joux-Lercier method to work over composite fields $\mathbb{F}_{p^n}$.

The heart of the GJL method is the following idea. Let $\phi(x)$ be a monic polynomial $\phi(x) = x^n + \phi_{n-1}x^{n-1} + \cdots + \phi_1 x + \phi_0$ and $r \geq \deg(\phi)$ be an integer. Let $n = \deg(\phi)$. Given $\phi(x)$ and r, define an $(r+1) \times (r+1)$ matrix $M_{\phi,r}$ in the following manner.

$$M_{\phi,r} = \begin{bmatrix} p & & & & & & \\ & \ddots & & & & & \\ & & \ddots & & & & \\ & & & p & & & \\ \phi_0 & \phi_1 & \cdots & \phi_{n-1} & 1 & & \\ & \ddots & \ddots & & & \ddots & \\ & & \phi_0 & \phi_1 & \cdots & \phi_{n-1} & 1 \end{bmatrix} \tag{14}$$

The first $n \times n$ principle sub-matrix of $M_{\phi,r}$ is $\mathrm{diag}[p, p, \ldots, p]$ corresponding to the polynomials $p, px, \ldots, px^{n-1}$. The last $r - n + 1$ rows correspond to the polynomials $\phi(x), x\phi(x), \ldots, x^{r-n}\phi(x)$.

Apply the LLL algorithm to $M_{\phi,r}$ and let the first row of the resulting LLL-reduced matrix be $[g_0, g_1, \ldots, g_{r-1}, g_r]$. Define

$$g(x) = g_0 + g_1 x + \cdots + g_{r-1} x^{r-1} + g_r x^r. \tag{15}$$

The notation

$$g = \mathrm{LLL}\left(M_{\phi,r}\right) \tag{16}$$

will be used to denote the polynomial $g(x)$ given by (15). By construction, $\phi(x)$ is a factor of $g(x)$ modulo p.

The GJL procedure for polynomial selection is the following. Choose an $r \geq n$ and repeat the following steps until f and g are irreducible over $\mathbb{Z}$ and ϕ is irreducible over $\mathbb{F}_p$.

1. Randomly choose a degree $(r+1)$-polynomial $f(x)$ which is irreducible over $\mathbb{Z}$ and having coefficients of size $O(\ln(p))$ such that $f(x)$ has a factor $\phi(x)$ of degree n modulo p which is both monic and irreducible.
2. Let $\phi(x) = x^n + \phi_{n-1}x^{n-1} + \cdots + \phi_1 x + \phi_0$ and $M_{\phi,r}$ be the $(r+1) \times (r+1)$ matrix given by (14).
3. Let $g(x) = \mathrm{LLL}\left(M_{\phi,r}\right)$.

The polynomial $f(x)$ has degree $r+1$ and $g(x)$ has degree r. The procedure is parameterised by the integer r.

The determinant of M is p^n and so from the properties of the LLL-reduced basis, the coefficients of $g(x)$ are of the order $O\left(p^{n/(r+1)}\right) = O\left(Q^{1/(r+1)}\right)$. The coefficients of $f(x)$ are $O(\ln(p))$.

The bound on the norm given by (13) in this case is $E^{2(2r+1)/t} Q^{(t-1)/(r+1)}$ which becomes $E^{2r+1} Q^{1/(r+1)}$ for $t = 2$. Increasing r reduces the size of the coefficients of $g(x)$ at the cost of increasing the degrees of f and g. In the concrete example considered in [4] and also in [21], r has been taken to be n and so M is an $(n+1) \times (n+1)$ matrix.

Conjugation. Repeat the following steps until f and g are irreducible over $\mathbb{Z}$ and ϕ is irreducible over $\mathbb{F}_p$.

1. Choose a quadratic monic polynomial $\mu(x)$, having coefficients of size $O(\ln p)$, which is irreducible over $\mathbb{Z}$ and has a root t in $\mathbb{F}_p$.
2. Choose two polynomials $g_0(x)$ and $g_1(x)$ with small coefficients such that $\deg g_1 < \deg g_0 = n$.
3. Let (u, v) be a rational reconstruction of t modulo p, i.e., $t \equiv u/v \bmod p$.
4. Define $g(x) = v g_0(x) + u g_1(x)$ and $f(x) = \mathrm{Res}_y(\mu(y), g_0(x) + y\, g_1(x))$.

Note that $\deg(f) = 2n$, $\deg(g) = n$, $\|f\|_\infty = O(\ln(p))$ and $\|g\|_\infty = O(p^{1/2}) = O(Q^{1/(2n)})$. In this case, the bound on the norm given by (13) is $E^{6n/t} Q^{(t-1)/(2n)}$ which becomes $E^{3n} Q^{1/(2n)}$ for $t = 2$.

4 A Simple Observation

For the GJL method, while constructing the matrix M, the coefficients of the polynomial $\phi(x)$ are used. If, however, some of these coefficients are zero, then these may be ignored. The idea is given by the following result.

Proposition 1. *Let n be an integer, d a divisor of n and $k = n/d$. Suppose $B(x)$ is a monic polynomial of degree k. Let $r \geq k$ be an integer and set $\psi(x) = \mathrm{LLL}(M_{B,r})$. Define $g(x) = \psi(x^d)$ and $\phi(x) = B(x^d)$. Then*

1. *$\deg(\phi) = n$ and $\deg(g) = rd$;*
2. *$\phi(x)$ is a factor of $g(x)$ modulo p;*
3. *$\|g\|_\infty = p^{n/(d(r+1))}$.*

Proof. For the first point, note that by construction $B(x)$ is a factor of $\psi(x)$ modulo p. So, $B(x^d)$ is a factor of $\psi(x^d) = g(x)$ modulo p. The coefficients of $g(x)$ are the coefficients of $\psi(x)$. Following the GJL method, $\|\psi\|_\infty = p^{k/(r+1)} = p^{n/(d(r+1))}$ and so the same holds for $\|g\|_\infty$. This shows the second point. □

Note that if we had defined $g(x) = \mathrm{LLL}(M_{\phi,rd})$, then $\|g\|_\infty$ would have been $p^{n/(rd+1)}$. For $d > 1$, the value of $\|g\|_\infty$ given by Proposition 1 is smaller.

A variant. The above idea shows how to avoid the zero coefficients of $\phi(x)$. A similar idea can be used to avoid the coefficients of $\phi(x)$ which are small. Suppose that the polynomial $\phi(x)$ can be written in the following form.

$$\phi(x) = \phi_{i_1} x^{i_1} + \cdots + \phi_{i_k} x^{i_k} + x^n + \sum_{j \notin \{i_1,\ldots,i_k\}} \phi_j x^j \tag{17}$$

where $i_1, \ldots, i_k$ are from the set $\{0, \ldots, n-1\}$ and for $j \in \{0, \ldots, n-1\} \setminus \{i_1, \ldots, i_k\}$, the coefficients ϕ_j are all $O(1)$. Some or even all of these ϕ_j's could be zero. A $(k+1) \times (k+1)$ matrix M is constructed in the following manner.

$$M = \begin{bmatrix} p & & & & \\ & \ddots & & & \\ & & \ddots & & \\ & & & p & \\ \phi_{i_1} & \phi_{i_2} & \cdots & \phi_{i_k} & 1 \end{bmatrix} \tag{18}$$

In the above the matrix M has only one row obtained from $\phi(x)$ and it is difficult to use more than one row. Apply the LLL algorithm to M and write the first row of the resulting LLL-reduced matrix as $[g_{i_1}, \ldots, g_{i_k}, g_n]$. Define

$$g(x) = (g_{i_1} x^{i_1} + \cdots + g_{i_k} x^{i_k} + g_n x^n) + \sum_{j \notin \{i_1,\ldots,i_k,n\}} \phi_j x^j. \tag{19}$$

The degree of $g(x)$ is n and the bound on the coefficients of $g(x)$ is determined as follows. The determinant of M is p^k and by the LLL-reduced property each of the coefficients $g_{i_1}, \ldots, g_{i_k}, g_n$ is $O(p^{k/(k+1)}) = O(Q^{k/(n(k+1))})$. Since ϕ_j for $j \notin \{i_1, \ldots, i_k\}$ are all $O(1)$, it follows from (19) that all the coefficients of $g(x)$ are $O(Q^{k/(n(k+1))})$ and so $\|g\|_\infty = O(Q^{k/(n(k+1))})$.

5 A New Polynomial Selection Method

In the simple observation made in the earlier section, the non-zero terms of the polynomial $g(x)$ are powers of x^d. This creates a restriction and does not turn out to be necessary to apply the main idea of the previous section. Once the polynomial $\psi(x)$ is obtained using the LLL method, it is possible to substitute any degree d polynomial with small coefficients for x and still the norm bound will hold. In fact, the idea can be expressed more generally in terms of resultants. Algorithm 1 below describes the new general method for polynomial selection.

Algorithm 1: A new method of polynomial selection.

Input: p, n, d (a factor of n) and $r \geq n/d$.
Output: $f(x)$, $g(x)$ and $\phi(x)$.

1.1 Let $k = n/d$;
1.2 **repeat**
1.3 Randomly choose a monic irreducible polynomial $A(x)$ having the following properties: $\deg(a) = r + 1$; $A(x)$ is irreducible over the integers; $A(x)$ has coefficients of size $O(\ln(p))$; modulo p, $A(x)$ has an irreducible factor $B(x)$ of degree k.
1.4 Randomly choose monic polynomials $C_0(x)$ and $C_1(x)$ with small coefficients such that $\deg C_0(x) = d$ and $\deg C_1(x) < d$.
1.5 Define

$$\begin{aligned} f(x) &= \mathrm{Res}_y\left(A(y), C_0(x) + y\, C_1(x)\right); \\ \phi(x) &= \mathrm{Res}_y\left(B(y), C_0(x) + y\, C_1(x)\right) \bmod p; \\ \psi(x) &= \mathrm{LLL}(M_{b,r}); \\ g(x) &= \mathrm{Res}_y\left(\psi(y), C_0(x) + y\, C_1(x)\right). \end{aligned}$$

1.6 **until** *$f(x)$ and $g(x)$ are irreducible over $\mathbb{Z}$ and $\phi(x)$ is irreducible over $\mathbb{F}_p$.*
1.7 **return** $f(x)$, $g(x)$ and $\phi(x)$.

The following result states the basic properties of Algorithm 1.

Proposition 2. *The outputs $f(x)$, $g(x)$ and $\phi(x)$ of Algorithm 1 satisfy the following.*

1. $\deg(f) = d(r+1)$; $\deg(g) = rd$ *and* $\deg(\phi) = n$;
2. *both $f(x)$ and $g(x)$ have $\phi(x)$ as a factor modulo p;*
3. $\|f\|_\infty = O(\ln(p))$ *and* $\|g\|_\infty = O(Q^{1/(d(r+1))})$.

Consequently,

$$\begin{aligned} |\mathrm{Res}(f,\lambda) \times \mathrm{Res}(g,\lambda)| &\approx \|f\|_\infty \|g\|_\infty \times E^{2(\deg f + \deg g)/t} \\ &= O\left(E^{2d(2r+1)/t} \times Q^{(t-1)/(d(r+1))}\right). \end{aligned} \tag{20}$$

Proof. By definition $f(x) = \mathrm{Res}_y\left(A(y), C_0(x) + y\,C_1(x)\right)$ where $A(x)$ has degree $r+1$, $C_0(x)$ has degree d and $C_1(x)$ has degree $d-1$, so the degree of $f(x)$ is $d(r+1)$. Similarly, one obtains the degree of $\phi(x)$ to be n. Since $\psi(x)$ is obtained from $B(x)$ as $\mathrm{LLL}(M_{b,r})$ it follows that the degree of $\psi(x)$ is r and so the degree of $g(x)$ is rd.

Since $B(x)$ divides $A(x)$ modulo p, it follows from the definition of $f(x)$ and $\phi(x)$ that modulo p, $\phi(x)$ divides $f(x)$. Since $\psi(x)$ is a linear combination of the rows of $M_{b,r}$, it follows that modulo p, $\psi(x)$ is a multiple of $B(x)$. So, $g(x) = \mathrm{Res}_y\left(\psi(y), C_0(x) + y\,C_1(x)\right)$ is a multiple of $\phi(x) = \mathrm{Res}_y\left(B(y), C_0(x) + y\,C_1(x)\right)$ modulo p.

Since the coefficients $C_0(x)$ and $C_1(x)$ are $O(1)$ and the coefficients $A(x)$ are $O(\ln p)$, it follows that $\|f\|_\infty = O(\ln p)$. The coefficients of $g(x)$ are $O(1)$ multiples of the coefficients of $\psi(x)$. By Proposition 1, the coefficients of $\psi(x)$ are $O(p^{n/(d(r+1))}) = Q^{1/(d(r+1))}$ which shows that $\|g\|_\infty = O(Q^{1/(d(r+1))})$. □

Proposition 2 provides the relevant bound on the product of the norms of a sieving polynomial $\lambda(x)$ in the two number fields defined by $f(x)$ and $g(x)$. We note the following points.

1. If $d = 1$, then the norm bound is $E^{2(2r+1)/t}Q^{(t-1)/(r+1)}$ which is the same as that obtained using the GJL method.
2. If $d = n$, then the norm bound is $E^{2n(2r+1)/t}Q^{(t-1)/(n(r+1))}$. Further, if $r = k = 1$, then the norm bound is the same as that obtained using the Conjugation method. So, for $d = n$, Algorithm 1 is a generalisation of the Conjugation method. Later, we show that choosing $r > 1$ provides asymptotic improvements.
3. If n is a prime, then the only values of d are either 1 or n. The norm bounds in these two cases are covered by the above two points.
4. If n is composite, then there are non-trivial values for d and it is possible to obtain new trade-offs in the norm bound. For concrete situations, this can be of interest. Further, for composite n, as value of d increases from $d = 1$ to $d = n$, the norm bound nicely interpolates between the norm bounds of the GJL method and the Conjugation method.

6 Comparison and Examples

We compare the norm estimates for $t = 2$, i.e., when the sieving polynomial is linear. In this case, Table 2 lists the degree and norm estimates of polynomials for various methods. Table 3 compares the new method with the JLSV1 and the GJL method for concrete values of n, r and d. This shows that the new method provides different trade-offs which were not known earlier.

As an example, we can see from Table 3 that the new method compares well with GJL and JLSV1 methods for $n = 4$ and $Q(dd) = 300$ (refer to Table 1). As mentioned in [4], when the differences between the methods are small, it is not possible to decide by looking only at the size of the norm product. Keeping this in view, we see that the new method is competetive for $n = 6$ as well. At

present there are no avaible estimates of E for the larger values of n. From the Q-E pairs given in Table 1, it is clear that the increase of E is slower than that of Q. This suggests that the new method will become competitive when Q is sufficiently large.

Table 2. Parameterised efficiency estimates for NFS obtained from the different polynomial selection methods.

Methods	$\deg f$	$\deg g$	$\|f\|_\infty$	$\|g\|_\infty$	$\|f\|_\infty\|g\|_\infty E^{(\deg f+\deg g)}$
JLSV1	n	n	$Q^{\frac{1}{2n}}$	$Q^{\frac{1}{2n}}$	$E^{2n}Q^{\frac{1}{n}}$
GJL ($r \geq n$)	$r+1$	r	$O(\ln p)$	$Q^{\frac{1}{r+1}}$	$E^{2r+1}Q^{\frac{1}{r+1}}$
Conjugation	$2n$	n	$O(\ln p)$	$Q^{\frac{1}{2n}}$	$E^{3n}Q^{\frac{1}{2n}}$
New method ($d\|n$, $r \geq n/d$)	$d(r+1)$	dr	$O(\ln p)$	$Q^{\frac{1}{d(r+1)}}$	$E^{d(2r+1)}Q^{1/(d(r+1))}$

Table 3. Comparison of efficiency estimates for composite n with $d = 2$ and $r = n/2$.

$\deg f, \deg g$	$\mathbb{F}_Q$	$\|f\|_\infty$	g	$\|g\|_\infty$	$\|f\|_\infty\|g\|_\infty E^{(\deg f+\deg g)}$
$(5,4)$	$\mathbb{F}_{p^4}$	$O(\ln p)$	GJL	$Q^{\frac{1}{5}}$	$E^9Q^{\frac{1}{5}}$
$(4,4)$		$Q^{\frac{1}{8}}$	JLSV1	$Q^{\frac{1}{8}}$	$E^8Q^{\frac{1}{4}}$
$(6,4)$		$O(\ln p)$	New	$Q^{\frac{1}{6}}$	$E^{10}Q^{\frac{1}{6}}$
$(7,6)$	$\mathbb{F}_{p^6}$	$O(\ln p)$	GJL	$Q^{\frac{1}{7}}$	$E^{13}Q^{\frac{1}{7}}$
$(6,6)$		$Q^{\frac{1}{12}}$	JLSV1	$Q^{\frac{1}{12}}$	$E^{12}Q^{\frac{1}{6}}$
$(8,6)$		$O(\ln p)$	New	$Q^{\frac{1}{8}}$	$E^{14}Q^{\frac{1}{8}}$
$(9,8)$	$\mathbb{F}_{p^8}$	$O(\ln p)$	GJL	$Q^{\frac{1}{9}}$	$E^{17}Q^{\frac{1}{9}}$
$(8,8)$		$Q^{\frac{1}{16}}$	JLSV1	$Q^{\frac{1}{16}}$	$E^{16}Q^{\frac{1}{8}}$
$(10,8)$		$O(\ln p)$	New	$Q^{\frac{1}{10}}$	$E^{18}Q^{\frac{1}{10}}$
$(10,9)$	$\mathbb{F}_{p^9}$	$O(\ln p)$	GJL	$Q^{\frac{1}{10}}$	$E^{19}Q^{\frac{1}{10}}$
$(9,9)$		$Q^{\frac{1}{18}}$	JLSV1	$Q^{\frac{1}{18}}$	$E^{18}Q^{\frac{1}{9}}$
$(12,9)$		$O(\ln p)$	New	$Q^{\frac{1}{12}}$	$E^{21}Q^{\frac{1}{12}}$

Next we provide some concrete examples of polynomials $f(x), g(x)$ and $\phi(x)$ obtained using the new method. The examples are for $n = 6$ and $n = 4$. For $n = 6$, we have taken $d = 1, 2, 3$ and 6 and in each case r was chosen to be $r = k = n/d$. For $n = 4$, we consider $d = 2$ with $r = k = n/d$ and $r = k + 1$; and $d = 4$ with $r = k$. These examples are to illustrate that the method works as predicted and returns the desired polynomials very fast. We have run the method

on other cases as well and have observed that it works fine in these other cases also.

Example 1. Let $n = 6$, and p is a 201-bit prime given below.

$$p = 1606938044258990275541962092341162602522202993782792835361211$$

Taking $d = 1$ and $r = d/n$, we get

$$f(x) = x^7 + 18\,x^6 + 99\,x^5 - 107\,x^4 - 3470\,x^3 - 15630\,x^2 - 30664\,x - 23239$$

$$\begin{aligned} g(x) = {} & 7129651367834661223841565542615046652356092434468 69\,x^6 \\ & +1604820385890326069176621670265257543528180754424771 2\,x^5 \\ & +1486772077481415492035898908528680282740771076248601 84\,x^4 \\ & +7240853845391439257955648357229262561719208529866603 72\,x^3 \\ & +1946932041954939829697950384964684583780249722185345772\,x^2 \\ & +2718971797270235171234259793142851416923331519178675874\,x \\ & +1517248296800681060244076172658712224507653769252953211 \end{aligned}$$

$$\begin{aligned} \phi(x) = {} & x^6 + 6715600759360122754018289503697292868061440059396953 \\ & 49290760\,x^5 + 7747058346245540667371991605555115020882703 \\ & 23481268337340514\,x^4 + 1100646447552671580437963861085020 \\ & 4311451261510579373184797 17\,x^3 + 2713164638641236582328 70 \\ & 0951132731200092664911740964726327 27\,x^2 + 410171738950673 \\ & 9512253510092562513530586956018743720805730 92\,x + 1326632 \\ & 8049610277672723346626935788558453638543982315243906 07 \end{aligned}$$

Note that $\|g\|_\infty \approx 2^{180}$. Taking $d = 2$ and $r = d/n$, we get

$$f(x) = x^8 - x^7 - 5\,x^6 - 50\,x^5 - 181\,x^4 - 442\,x^3 - 801\,x^2 - 633\,x - 787$$

$$\begin{aligned} g(x) = {} & 8334809325005164925059358391850081936964577 87\,x^6 + 209259 \\ & 3616641287655065740032896986343580698615\,x^5 + 12985408995 \\ & 6895226179153774346833519431885333 20\,x^4 + 218697415909663 \\ & 5789762016746153996714153297062 2\,x^3 + 6440309722463426267 \\ & 7273803471992671747860968564\,x^2 + 5586471169528158428390 9 \\ & 4556655210927495027938 07\,x + 9217783540590778272527843567 \\ & 0487132710722661831 \end{aligned}$$

$$\begin{aligned}\phi(x) = x^6 &+ 22557756689804128540553922618322150822628658922554 \\ &6142714057\,x^5 + 72615667372388908289535145173973354532839 \\ &4720523246272955173\,x^4 + 10214781320546947215788889940017 \\ &3076493445466063054368834805 6\,x^3 + 67497810255620874288201 \\ &8027719951308454078609348118158783 91\,x^2 + 632426210761786 \\ &6221054941943149378179274393729180290427188 43\,x + 1040935 \\ &3068660167025264551437254153796047423390654217938440 38\end{aligned}$$

Note that $\|g\|_\infty \approx 2^{156}$. Taking $d = 3$ and $r = d/n$, we get

$$f(x) = x^9 - 4\,x^8 - 54\,x^7 - 174\,x^6 - 252\,x^5 - 174\,x^4 - 76\,x^3 - 86\,x^2 - 96\,x - 42$$

$$\begin{aligned}g(x) = &28897423645083815575933123924978010067 12\,x^6 + 8363369537 \\ &0646306085610877651462747385 09\,x^5 + 108280788065240857055 \\ &0641278340877294187 7\,x^4 + 4181282488973040016900039741726 \\ &7197701179\,x^3 + 14974213477775324762133150889796948238735 \\ &4\,x^2 + 2409467169894432102934429655526113055921 94\,x + 1516 \\ &9645565510474440307374333394042659883 3\end{aligned}$$

$$\begin{aligned}\phi(x) = x^6 &+ 26507457770597862491534287197053834813201015436810 9 \\ &244143774\,x^5 + 21159801273629654486978970226092134077566 6 \\ &75973129512551886\,x^4 + 10634456114456842669412895408279 47 \\ &1993974162763341880558378 92\,x^3 + 14595872830580543656399 5 \\ &0761731919998074021438242745336103973\,x^2 + 14565434378005 \\ &7164332563864820718837111792353916826321052299 5\,x + 37812 \\ &9170960510211491600303623674471468414144797178846977007\end{aligned}$$

Note that $\|g\|_\infty \approx 2^{137}$. Taking $d = 6$ and $r = d/n$, we get

$$\begin{aligned}f(x) = x^{12} &+ 3\,x^{10} + 10\,x^9 + 53\,x^8 + 112\,x^7 + 163\,x^6 \\ &+184\,x^5 + 177\,x^4 + 166\,x^3 + 103\,x^2 + 72\,x + 48\end{aligned}$$

$$\begin{aligned}g(x) = &-66687813840235319549883266984 8\,x^6 - 18672532710749247460 \\ &11849188889\,x^5 - 56017598132247742380355475666 67\,x^4 - 66687 \\ &5380176521094806391526505 3\,x^3 - 42680035364200678470378822 \\ &26971\,x^2 - 69355160900294806290332129063 63\,x - 746901308429 \\ &9698984047396755556\end{aligned}$$

$$\phi(x) = x^6 + 3564853368470740919209445971878112844118490479913342661856841 x^5 + 1069456010541222275762833791563433853235547143974002798557052 x^4 + 1754886399763801840627608935978938195370422461738784955672053 x^3 + 1069456010541222275762833791563433853235547143974002798557050 x^2 + 1069456010541222275762833791563433853235547143974002798557054 x + 1425941347388296367683778388751245137647396191965337064742736$$

In this case we get $\|g\|_\infty \approx 2^{102}$.

Example 2. Let $n = 4$, and p is a 301-bit prime given below.

$$p = 2037035976334486086268445688409378161051468393665936250636140449354381299763336706183493607$$

Taking $d = 2$ and $r = n/d$, we get

$$f(x) = x^6 + 2\,x^5 + 10\,x^4 + 11\,x^3 + 8\,x^2 + 3\,x + 5$$

$$g(x) = 1108486244023576208689360410176300373132220654590976786482134\,x^4 + 2050762938144982289360096083705563965935573667103554994528044\,x^3 + 5523467580377021934753091786207648479867036209679151793015319\,x^2 + 4562227246514756745388645848004531501269616133890841445574058\,x + 44149813363534457260637313760313481067348155550881750065331853$$

$$\phi(x) = x^4 + 1305623360698284685175599277707343457576279146188242586245210199344777856138293049165536292\,x^3 + 1630663764713242722426772175575945319640664\,x^2 + 1955704168728200759677945073444547181705052165401683279062058892036363498367414896214457800\,x + 1630663764713242722426772175575945319640665655794962932653634545690570677252853972689997047$$

In this case we have $\|g\|_\infty \approx 2^{201}$. If we take $r = n/d + 1$, we get

$$f(x) = x^8 + 16\,x^7 + 108\,x^6 + 398\,x^5 + 865\,x^4 + 1106\,x^3 + 820\,x^2 + 328\,x + 55$$

$$g(x) = 348482147842083865380881347784399925335728557\,x^6 + 553610 \\ 397998221059018601644545928977302904561 8\,x^5 + 33812545050 \\ 706664774530525723335145801290667783\,x^4 + 960621719572611 \\ 247634285906489587458873544533 0\,x^3 + 1240857957813073637 \\ 599358981318875637925354890 69\,x^2 + 7309083997372916996696 \\ 406142840231613191113080 8\,x + 160938107832743090553504819 \\ 72028841649178007790$$

$$\phi(x) = x^4 + 512869096459794324650196235899867623703393084616896 \\ 744799033424455696319185673262765599428\,x^3 + 180240879693 \\ 274948744497479057602208170834465922920791127184582765 00 \\ 357133832684276624164 44\,x^2 + 155334120802632167628916463 7 \\ 552573668603116979990828843347557957477286150023843804 26 \\ 2435184\,x + 263801507553366513494386082876419210598165405 \\ 37851767687474555428294675582624863936561816 8$$

In this case we have $\|g\|_\infty \approx 2^{156}$. If we take $d = 4$ and $r = d/n$, we have

$$f(x) = x^8 - 3\,x^7 - 33\,x^6 - 97\,x^5 - 101\,x^4 + 3\,x^3 + 73\,x^2 - 35\,x - 8$$

$$g(x) = 6848628860241259739113918671984158414368772 78\,x^4 + 192580 \\ 839295706051924893370529558897477491073 1\,x^3 + 16682478627 \\ 264257142784499126962718757034685 25\,x^2 + 4096156044753896 \\ 148518238570012309375827176 3\,x + 124094550693293454533754 \\ 1838097173133338033453$$

$$\phi(x) = x^4 + 300129299129056665818770804611316232682274696357657 \\ 624805901338072170670924524605598965 54\,x^3 + 9003878973871 \\ 699974563124138339486980468240890729728744174 0401421651 2 \\ 01277357381679689656\,x^2 + 150064649564528332909385402305 6 \\ 581163411373481788288124029506690360853354622623027994 82 \\ 756\,x + 300129299129056665818770804611316232682274696357 6 \\ 576248059013380721706709245246055989655 3$$

In this case also we have $\|g\|_\infty \approx 2^{150}$.

7 Asymptotic Complexity Analysis

Recall that $Q = p^n$. The subexponential expression $L_Q(a, c)$ is defined to be the following.

$$L_Q(a, c) = \exp\left((c + o(1))(\ln Q)^a (\ln \ln Q)^{1-a}\right). \tag{21}$$

The goal of the asymptotic complexity analysis is to express the runtime of the NFS algorithm using the L-notation and at the same time obtain bounds on p for which the analysis is valid. Our description of the analysis is based on prior works predominantly those in [13, 17, 4, 21].

For $0 < a < 1$, write

$$p = L_Q(a, c_p), \text{ where } c_p = \frac{1}{n}\left(\frac{\ln Q}{\ln\ln Q}\right)^{1-a} \text{ and so } n = \frac{1}{c_p}\left(\frac{\ln Q}{\ln\ln Q}\right)^{1-a} \quad (22)$$

The value of a will be determined later. Also, for each c_p, the runtime of the NFS algorithm is the same for the family of finite fields $\mathbb{F}_{p^n}$ where p is given by (22).

From Section 2, we recall the following.

1. The number of polynomials to be considered for sieving is E^2.
2. The factor base is of size $2B$ and we ignore the constant 2 for the asymptotic analysis.

Sparse linear algebra using the Lanczos or the block Wiedemann algorithm takes time $O(B^2)$. For some $0 < b < 1$, let

$$B = L_Q(b, c_b). \quad (23)$$

The value of b will be determined later. Set

$$E = B \quad (24)$$

so that the number of sieving polynomial is equal to the time for the linear algebra.

The bound on the product of the norms given by Proposition 2 is

$$\Gamma = E^{\frac{2}{t}d(2r+1)} \times Q^{\frac{t-1}{d(r+1)}}. \quad (25)$$

Suppose that Γ can be written as

$$\Gamma = L_Q(z, \zeta). \quad (26)$$

Using the L-notation version of the Canfield-Erdös-Pomerance theorem, the probability π that integers at most $\Gamma = L_Q(z, \zeta)$ are B-smooth with $B = L_Q(b, c_b)$ is given by

$$\pi^{-1} = L_Q\left(z - b, (z - b)\frac{\zeta}{c_b}\right). \quad (27)$$

Following the usual convention, we assume that the same smoothness probability holds for the event that a random sieving polynomial $\lambda(x)$ is smooth over the factor base.

The expected number of polynomials to consider for obtaining one relation is π^{-1}. Since B relations are required, obtaining this number of relations requires

trying $B\pi^{-1}$ trials. Balancing the cost of sieving and the linear algebra steps requires $B\pi^{-1} = B^2$ and so

$$\pi^{-1} = B. \tag{28}$$

Obtaining π^{-1} from (27) and setting it to be equal to B allows solving for c_b. Balancing the costs of the sieving and the linear algebra phases leads to the runtime of the NFS algorithm to be $B^2 = L_Q(b, 2c_b)$. So, to determine the runtime, we need to determine b and c_b. The value of b will turn out to be $1/3$ and the only real issue is the value of c_b.

Lemma 1. *Let $n = kd$ for positive integers k and d. Using the expressions for p and $E(= B)$ given by (22) and (23), we obtain the following.*

$$\left.\begin{aligned} E^{\frac{2}{t}d(2r+1)} &= L_Q\left(1-a+b, \tfrac{2c_b(2r+1)}{c_pkt}\right); \\ Q^{\frac{t-1}{d(r+1)}} &= L_Q\left(a, \tfrac{kc_p(t-1)}{(r+1)}\right). \end{aligned}\right\} \tag{29}$$

Proof. The second expression follows directly from $Q = p^n$, $p = L_Q(a, c_p)$ and $n = kd$. The computation for obtaining the first expression is the following.

$$\begin{aligned} E^{\frac{2}{t}d(2r+1)} &= L_Q\left(b, c_b\frac{2}{t}d(2r+1)\right) \\ &= \exp\left(c_b\frac{2}{t}(2r+1)\frac{n}{k}(\ln Q)^b(\ln\ln Q)^{1-b}\right) \\ &= \exp\left(c_b\frac{2}{c_pkt}(2r+1)\left(\frac{\ln Q}{\ln\ln Q}\right)^{1-a}(\ln Q)^b(\ln\ln Q)^{1-b}\right) \\ &= L_Q\left(1-a+b, \frac{2c_b(2r+1)}{c_pkt}\right). \end{aligned}$$

□

Theorem 1 (Boundary Case). *Let k divide n, $r \geq k$, $t \geq 2$ and $p = L_Q(2/3, c_p)$ for some $0 < c_p < 1$. It is possible to ensure that the runtime of the NFS algorithm with polynomials chosen by Algorithm 1 is $L_Q(1/3, 2c_b)$ where*

$$c_b = \frac{2r+1}{3c_pkt} + \sqrt{\left(\frac{2r+1}{3c_pkt}\right)^2 + \frac{kc_p(t-1)}{3(r+1)}}. \tag{30}$$

Proof. Setting $2a = 1 + b$, the two L-expressions given by (29) have the same first component and so the product of the norms is

$$\Gamma = L_Q\left(a, \frac{2c_b(2r+1)}{c_pkt} + \frac{kc_p(t-1)}{(r+1)}\right).$$

Then π^{-1} given by (27) is

$$L_Q\left(a-b, (a-b)\left(\frac{2(2r+1)}{c_pkt} + \frac{kc_p(t-1)}{c_b(r+1)}\right)\right).$$

From the condition $\pi^{-1} = B$, we get

$$\begin{aligned} b &= a - b; \\ c_b &= (a-b)\left(\frac{2(2r+1)}{c_p k t} + \frac{k c_p (t-1)}{c_b (r+1)}\right). \end{aligned}$$

The conditions $a - b = b$ and $2a = 1 + b$ show that $b = 1/3$ and $a = 2/3$. The second equation then becomes

$$c_b = \frac{1}{3}\left(\frac{2(2r+1)}{c_p k t} + \frac{k c_p (t-1)}{c_b (r+1)}\right). \tag{31}$$

Solving the quadratic for c_b and choosing the positive root gives

$$c_b = \frac{2r+1}{3 c_p k t} + \sqrt{\left(\frac{2r+1}{3 c_p k t}\right)^2 + \frac{k c_p (t-1)}{3(r+1)}}.$$

□

The boundary case complexity of the conjugation method is obtained as a special case of Theorem 1.

Corollary 1 (Boundary Case of the Conjugation Method [4]). *Let $r = k = 1$. Then for $p = L_Q(2/3, c_p)$, the runtime of the NFS algorithm is $L_Q(1/3, 2c_b)$ with*

$$c_b = \frac{1}{c_p t} + \sqrt{\left(\frac{1}{c_p t}\right)^2 + \frac{c_p (t-1)}{6}}.$$

Allowing r to be greater than k leads to improved asymptotic complexity. We do not perform this analysis. Instead, we perform the analysis in the similar situation which arises for the multiple number field sieve algorithm.

Theorem 2 (Medium Characteristic Case). *Let $p = L_Q(a, c_p)$ with $a > 1/3$. It is possible to ensure that the runtime of the NFS algorithm with the polynomials produced by Algorithm 1 is $L_Q(1/3, (32/3)^{1/3})$.*

Proof. Since $a > 1/3$, the bound Γ on the product of the norms can be taken to be the expression given by (13).

The parameter t is chosen as follows. For $0 < c < 1$, let

$$t = c_t n \left(\frac{\ln Q}{\ln \ln Q}\right)^{-c}. \tag{32}$$

Then the expressions given by (29) become the following.

$$\left.\begin{aligned} E^{\frac{2}{t} d(2r+1)} &= L_Q\left(b + c, \tfrac{2 c_b (2r+1)}{k c_t}\right); \\ Q^{\frac{t-1}{d(r+1)}} &= L_Q\left(1 - c, \tfrac{k c_t}{r+1}\right). \end{aligned}\right\} \tag{33}$$

This can be seen by substituting the expression for t in (29) and further by using the expression for n given in (22). Note that for the asymptotic analysis, $t-1$ is also assumed to be given by (32).

Setting $2c = 1-b$, the first components of the two expressions in (33) become equal and so

$$\Gamma = L_Q\left(b+c, \frac{2c_b(2r+1)}{kc_t} + \frac{kc_t}{r+1}\right).$$

Using this Γ, the expression for π^{-1} is

$$\pi^{-1} = L_Q\left(c, c\left(\frac{2(2r+1)}{kc_t} + \frac{kc_t}{c_b(r+1)}\right)\right).$$

We wish to choose c_t so as to maximise the probability π and hence to minimise π^{-1}. This is done by setting

$$\frac{2(2r+1)}{kc_t} = \frac{kc_t}{c_b(r+1)}$$

whence $kc_t = \sqrt{2c_b(r+1)(2r+1)}$. With this value of kc_t,

$$\pi^{-1} = L_Q\left(c, \frac{2c\sqrt{2c_b(r+1)(2r+1)}}{c_b(r+1)}\right).$$

Setting π^{-1} to be equal to $B = L_Q(b, c_b)$ yields

$$\begin{aligned} b &= c; \\ c_b &= \left(\frac{2c\sqrt{2c_b(r+1)(2r+1)}}{c_b(r+1)}\right). \end{aligned}$$

From $b = c$ and $2c = 1-b$ we obtain $c = b = 1/3$. Using this value of c in the equation for c_b, we obtain

$$c_b = \left(\frac{2}{3}\right)^{2/3} \times \left(\frac{2(2r+1)}{r+1}\right)^{1/3}.$$

The value of c_b is the minimum for $r = 1$ and this value is $c_b = (4/3)^{1/3}$. □

Note that the parameter a which determines the size of p is not involved in any of the computation. The assumption $a > 1/3$ is require to ensure that the bound on the product of the norms can be taken to be the expression given by (13). The proof of the theorem shows that the generality introduced by k and r does not affect the overall asymptotic complexity which is the same as that obtained in [4].

Theorem 3 (Large Characteristic). *It is possible to ensure that the runtime of the NFS algorithm with the polynomials produced by Algorithm 1 is* $L_Q(1/3, (64/9)^{1/3})$ *for* $p \geq L_Q(2/3, (8/3)^{1/3})$.

Proof. For $0 < e < 1$, let

$$r = \frac{c_r}{2}\left(\frac{\ln Q}{\ln\ln Q}\right)^e. \tag{34}$$

For the asymptotic analysis, the expression for $2r+1$ is taken to be two times the expression given by (34). Substituting this expression for r in (29), we obtain

$$\left.\begin{aligned} E^{\frac{2}{t}d(2r+1)} &= L_Q\left(1-a+b+e, \tfrac{2c_bc_r}{c_pkt}\right); \\ Q^{\frac{t-1}{d(r+1)}} &= L_Q\left(a-e, \tfrac{2kc_p(t-1)}{c_r}\right). \end{aligned}\right\} \tag{35}$$

Setting $1+b = 2(a-e)$, we obtain

$$\Gamma = L_Q\left(\frac{1+b}{2}, \frac{2c_bc_r}{c_pkt} + \frac{2kc_p(t-1)}{c_r}\right)$$

and so the probability π^{-1} is given by

$$L_Q\left(\frac{1-b}{2}, \frac{1-b}{2} \times \left(\frac{2c_r}{c_pkt} + \frac{2kc_p(t-1)}{c_rc_b}\right)\right).$$

The choice of c_r for which the probability π is maximised (and hence π^{-1} is minimised) is obtained by setting

$$\frac{c_r}{c_pk} = \sqrt{\frac{t(t-1)}{c_b}}$$

and the minimum value of π^{-1} is

$$L_Q\left(\frac{1-b}{2}, \frac{1-b}{2} \times \left(4\sqrt{\frac{t-1}{tc_b}}\right)\right).$$

Setting this value of π^{-1} to be equal to B, we obtain

$$\begin{aligned} b &= (1-b)/2; \\ c_b &= \frac{1-b}{2} \times \left(4\sqrt{\frac{t-1}{tc_b}}\right). \end{aligned}$$

The first equation shows $b = 1/3$ and using this in the second equation, we obtain $c_b = (4/3)^{2/3}((t-1)/t)^{1/3}$. This value of c_b is minimised for the minimum value of t which is $t = 2$. This gives $c_b = (8/9)^{1/3}$.

Using $2(a-e) = 1+b$ and $b = 1/3$ we get $a - e = 2/3$. Note that $r \geq k$ and so $p \geq p^{k/r} = L_Q(a, (c_pk)/r) = L_Q(a-e, (2c_pk)/c_r)$. With $t = 2$, the value of $(c_pk)/c_r$ is equal to $(1/3)^{1/3}$ and so $p \geq L_Q(2/3, (8/3)^{1/3})$. □

8 Multiple Number Field Sieve Variant

As the name indicates, the multiple number field sieve variant uses several number fields. The discussion and the analysis will follow the works [6, 21].

In the single number field sieve algorithm, there are two number fields (say, the left and the right) and the sieving polynomial $\phi(x)$ is to satisfy smoothness conditions on both of these. In the multiple number field sieve variant, the left number field remains unchanged. The right number field is replaced by a collection of V number fields. The sieving polynomial $\phi(x)$ has to satisfy the smoothness condition on the left number field as before. On the right side, it is sufficient for $\phi(x)$ to satisfy a smoothness condition on at least one of the number fields.

Recall that the Algorithm 1 produces two polynomials $f(x)$ and $g(x)$ of degrees $d(r+1)$ and dr respectively. The polynomial $g(x)$ is defined to be $\mathrm{Res}_y(\psi(y), c_0(x) + yc_1(x))$ where $\psi(x) = \mathrm{LLL}(M_{B,r})$, i.e., $\psi(x)$ is defined from the first row of the matrix obtained after applying the LLL-algorithm to $M_{b,r}$.

Methods for obtaining the collection of number fields on the right has been mentioned in [21]. We adapt one of these methods to our setting. Consider Algorithm 1. Let $\psi_1(x)$ be $\psi(x)$ as above and let $\psi_2(x)$ be the polynomial defined from the second row of the matrix $M_{B,r}$. Define $g_1(x) = \mathrm{Res}_y(\psi_1(y), c_0(x) + yc_1(x))$ and $g_2(x) = \mathrm{Res}_y(\psi_2(y), c_0(x) + yc_1(x))$. Then choose $V-2$ linear combinations $g_i(x) = s_i g_1(x) + t_i g_2(x)$, for $i = 3, \ldots, V$. All the g_i's have degree dr. Asymptotically, $\|\psi_2\|_\infty = \|\psi_1\|_\infty = Q^{1/(d(r+1))}$. So, all the g_i's have ther L_∞ norms to be the same as that of $g(x)$ given by Proposition 2.

For the left number field, as before let B be the bound on the norms of the ideals which are in the factor basis defined by f. For each of the right number fields, let B' be the bound on the norms of the ideals which are in the factor basis defined by each of the g_i's. So, the size of the entire factor basis is $B + VB'$. The following condition balances the left portion and the right portion of the factor basis.

$$B = VB'. \tag{36}$$

With this condition, the size of the factor basis is $2B$ and so linear algebra takes time B^2 after ignoring the constant 4. As before, the number of sieving polynomials is $E^2 = B^2$ and the coefficients of $\phi(x)$ can take $E^{2/t}$ distinct values.

Let $\Psi(\Gamma, B)$ be the probability that a random positive integer which is at most Γ is B-smooth. As noted earlier, this probability is given by the Canfield-Erdös-Pomerance theorem. Further, if Γ and B are expressed in the L-notation, then $\Psi(\Gamma, B)$ can also be expressed in the L-notation.

Let π be the probability that a random sieving polynomial $\phi(x)$ gives rise to a relation. Let π_1 be the probability that $\phi(x)$ is smooth over the left factor basis and π_2 be the probability that $\phi(x)$ is smooth over *at least* one of the right factor bases. Further, let $\Gamma_1 = \mathrm{Res}_x(f(x), \phi(x))$ be the bound on the norm corresponding to the left number field and $\Gamma_2 = \mathrm{Res}_x(g_i(x), \phi(x))$ be the bound on the norm for any of the right number fields. Note that Γ_2 is determined only

by the degree and the L_∞-norm of $g_i(x)$ and hence is the same for all $g_i(x)$'s. Heuristically, we have

$$\begin{aligned} \pi_1 &= \Psi(\Gamma_1, B); \\ \pi_2 &= V\Psi(\Gamma_2, B'); \\ \pi &= \pi_1 \times \pi_2. \end{aligned} \tag{37}$$

As before, one relation is obtained in about π^{-1} trials and so B relations are obtained in about $B\pi^{-1}$ trials. Balancing the cost of linear algebra and sieving, we have as before $B = \pi^{-1}$.

The following choices of B and V are made.

$$\begin{aligned} E = B &= L_Q\left(\tfrac{1}{3}, c_b\right); \\ V &= L_Q\left(\tfrac{1}{3}, c_v\right); \text{ and so} \\ B' &= B/V = L_Q\left(\tfrac{1}{3}, c_b - c_v\right). \end{aligned} \tag{38}$$

With these choices of B and V, it is possible to analyse the MNFS variant for Algorithm 1 for three cases, namely, the medium prime case, the boundary case and the large characteristic case. Below we present the details of the boundary case. This presents a new asymptotic result.

Theorem 4 (MNFS-Boundary Case). *Let k divide n, $r \geq k$, $t \geq 2$ and*

$$p = L_Q\left(\frac{2}{3}, c_p\right) \text{ where } c_p = \frac{1}{n}\left(\frac{\ln Q}{\ln\ln Q}\right)^{1/3}.$$

It is possible to ensure that the runtime of the MNFS algorithm is $L_Q(1/3, 2c_b)$ where

$$c_b = \frac{2(r+1)}{3ktc_p} - c_v + \frac{1}{3(c_b - c_v)}\left(\frac{2rc_b}{c_p kt} + \frac{kc_p(t-1)}{r+1}\right). \tag{39}$$

Proof. First note the following computations.

$$\begin{aligned}
\Gamma_1 &= \|\phi\|_\infty^{\deg(f)} = E^{2\deg(f)/t} = E^{(2d(r+1))/t} = E^{(2n(r+1))/(kt)} \\
&= L_Q\left(\frac{1}{3}, \frac{2(r+1)c_b}{ktc_p}\right); \\
\pi_1^{-1} &= L_Q\left(\frac{1}{3}, \frac{2(r+1)}{3ktc_p}\right); \\
\Gamma_2 &= \|\phi\|_\infty^{\deg(g)} \times \|g\|_\infty^{\deg(\phi)} \\
&= E^{2\deg(g)/t} \times Q^{(t-1)/(d(r+1))} \\
&= E^{(2rd)/t} \times Q^{(t-1)/(d(r+1))} \\
&= E^{(2rn)/(kt)} \times Q^{k(t-1)/(n(r+1))} \\
&= L_Q\left(\frac{1}{3}, \frac{2rc_b}{c_pkt} + \frac{kc_p(t-1)}{r+1}\right); \\
\pi_2^{-1} &= L_Q\left(\frac{1}{3}, -c_v + \frac{1}{3(c_b-c_v)}\left(\frac{2rc_b}{c_pkt} + \frac{kc_p(t-1)}{r+1}\right)\right); \\
\pi^{-1} &= L_Q\left(\frac{1}{3}, \frac{2(r+1)}{3ktc_p} - c_v + \frac{1}{3(c_b-c_v)}\left(\frac{2rc_b}{c_pkt} + \frac{kc_p(t-1)}{r+1}\right)\right);
\end{aligned}$$

From the condition $\pi^{-1} = c_b$, we obtain the following equation.

$$c_b = \frac{2(r+1)}{3ktc_p} - c_v + \frac{1}{3(c_b-c_v)}\left(\frac{2rc_b}{c_pkt} + \frac{kc_p(t-1)}{r+1}\right). \tag{40}$$

We wish to find c_v such that c_b is minimised subject to the constraint (40). Using the method of Lagrange multipliers, the partial derivative of (40) with respect to c_v gives

$$c_v = \frac{r+1}{3ktc_p}.$$

Using this value of c_v in (40) provides the following quadratic in c_b.

$$(3ktc_p)c_b^2 - (4r+2)c_b + \frac{(r+1)^2}{3ktc_p} - \frac{(c_pk)^2t(t-1)}{r+1} = 0.$$

Solving this and taking the positive square root, we obtain

$$c_b = \frac{4r+2}{6ktc_p} + \sqrt{\frac{r(3r+2)}{(3ktc_p)^2} + \frac{c_pk(t-1)}{3(r+1)}}. \tag{41}$$

Hence the overall complexity of MNFS for the boundary case is $L_Q\left(\frac{1}{3}, 2c_b\right)$. □

8.1 Further Analysis of the Boundary Case

In Figure 2, we have plotted $2c_b$ given by Theorem 4 against c_p for some values of t, k and r. The plot is labelled MNFS-New. The figure also shows the plot

of NFS-New which shows the complexity in the boundary case given by Theorem 1. For comparison, we have plotted the complexities of the GJL and the Conjugation methods from [4] and the MNFS-GJL and the MNFS-Conjugation methods from [21].

Based on these plots, we make the following observations.

1. Complexities of NFS-New are never worse than the complexities of NFS-GJL and NFS-Conj. Similarly, complexities of MNFS-New are never worse than the complexities of MNFS-GJL and MNFS-Conj.
2. For both the NFS-New and the MNFS-New methods, increasing the value of r provides new complexity trade-offs.
3. There is a value of c_p for which the minimum complexity is achieved. This corresponds to the MNFS-Conj. Let $L_Q(1/3, \theta_0)$ be this complexity. The value of θ_0 is determined later[2]
4. Let the complexity of the MNFS-GJL be $L_Q(1/3, \theta_1)$. The value of θ_1 was determined in [21]. The plot for MNFS-New approaches the plot for MNFS-GJL from below.
5. For smaller values of c_p, it is advantageous to choose $t > 2$ and also $k > 1$. On the other hand, for larger values of c_p, the minimum complexity is attained for $t = 2$ and $k = 1$.

From the plot, it can be seen that for larger values of c_p, the minimum value of c_b is attained for $t = 2$ and $k = 1$. So, we decided to perform further analysis using these values of t and k. Let

$$C = 2c_b = 2\sqrt{\frac{c_p}{3\,(r+1)} + \frac{(3\,r+2)r}{36\,c_p^2}} + \frac{2\,r+1}{3\,c_p} \tag{42}$$

Note that C is a function of both c_p and r. For each r, we wish to find the value of c_p for which C is minimised. Let us denote this value as $C(r)$. Our aim is to show that $\theta_0 = C(1)$; $C(r)$ is monotone increasing for $r \geq 1$; and the limiting upper bound of $C(r)$ is θ_1. The analysis below is to establish these facts.

Differentiating C with respect to c_p and equating to 0 gives

$$\frac{\frac{6}{r+1} - \frac{(3\,r+2)r}{c_p^3}}{18\sqrt{\frac{c_p}{3\,(r+1)} + \frac{(3\,r+2)r}{36\,c_p^2}}} - \frac{2\,r+1}{3\,c_p^2} = 0 \tag{43}$$

On simplifying we get,

$$\frac{6c_p^3 - (3r+2)r(r+1)}{\sqrt{\Big(12c_p^3 + (r+1)(3r+2)r\Big)(r+1)}} - \frac{2\,r+1}{1} = 0 \tag{44}$$

[2] Due to an error in calculation, the value of θ_0 determined in [21] was incorrect.

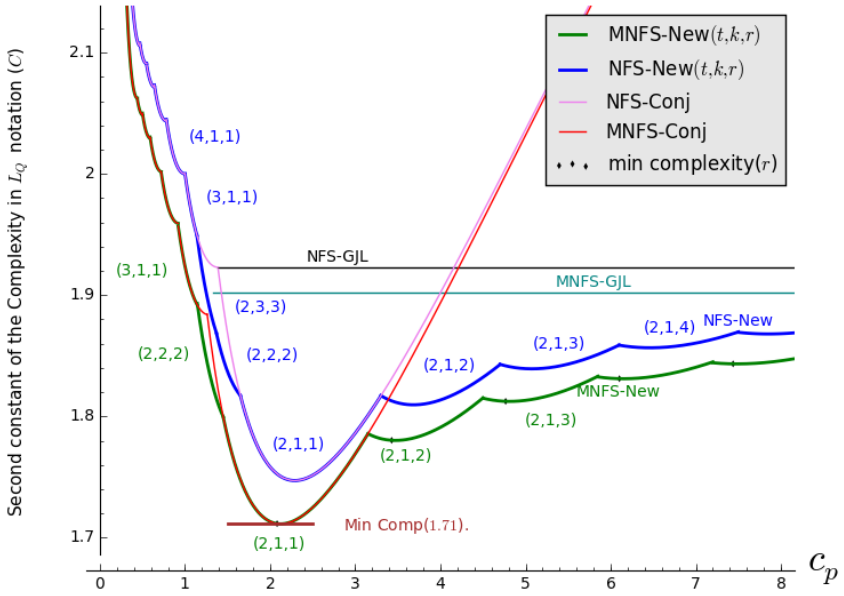

Fig. 2. Complexity plot for boundary case

Equation (44) is quadratic in c_p^3. On solving we get the following value of c_p.

$$\begin{aligned} c_p &= \left(\frac{7}{6}\,r^3 + 2\,r^2 + \frac{1}{6}\,\sqrt{13\,r^2+8\,r+1}\,(2\,r^2+3\,r+1) + r + \frac{1}{6}\right)^{1/3} \\ &= \rho(r) \ \text{(say)} \end{aligned} \tag{45}$$

Putting the value of c_p back in (42), we get the minimum value of C (in terms of r) as

$$C(r) = 2\,\sqrt{\frac{\rho(r)}{3\,(r+1)} + \frac{(3\,r+2)r}{36\,\rho(r)^2} + \frac{2\,r+1}{3\,\rho(r)}}. \tag{46}$$

All the three sequences in the expression for $C(r)$, viz, $\frac{\rho(r)}{3\,(r+1)}$, $\frac{(3\,r+2)r}{36\,\rho(r)^2}$ and $\frac{2\,r+1}{3\,\rho(r)}$ are monotonic increasing. This can be verified through computation (with a symbolic algebra package) as follows. Let s_r be any one of these sequences. Then computing s_{r+1}/s_r gives a ratio of polynomial expressions from which it is possible to directly argue that s_{r+1}/s_r is greater than one. We have done these computations but, do not present the details since they are uninteresting and quite messy. Since all the three sequences $\frac{\rho(r)}{3\,(r+1)}$, $\frac{(3\,r+2)r}{36\,\rho(r)^2}$ and $\frac{2\,r+1}{3\,\rho(r)}$ are monotonic increasing so is $C(r)$.

Note that for $r \geq 1$,

$$\rho(r) > \left(\frac{7}{6}\right)^{1/3} r, \quad \text{i.e.,} \quad \rho(r) > 1.05r. \tag{47}$$

So, for $r > 1$,

$$\frac{(3r+2)r}{\rho(r)^2} = 3\left(\frac{r}{\rho(r)}\right)^2 + 2\frac{r}{\rho(r)^2} < 3 \times \left(\frac{1}{1.05}\right)^2 + 2 \times \frac{1}{1.05}.$$
$$\frac{(2r+1)}{\rho(r)} = 2\frac{r}{\rho(r)} + \frac{1}{\rho(r)} < 2 \times \frac{1}{1.05} + \frac{1}{1.05}.$$

This shows that the sequences $\frac{(3r+2)r}{\rho(r)^2}$ and $\frac{(2r+1)}{\rho(r)}$ are bounded above. For $r > 8$, we have

$$(3r+1) < (8r+1) < r^2 \text{ and } \left(2r^2 + r + \frac{1}{6}\right) < \frac{r^3}{3}$$

which implies that

$$\rho(r) < \left(\frac{7}{6} + \frac{1}{6} \times \sqrt{14} \times 3 + \frac{1}{3}\right)^{1/3} r < 1.5\, r \text{ for } r > 8. \tag{48}$$

Using $\rho(r) < 1.5r$ for $r > 8$, it can be shown that the sequence $\left(\frac{\rho(r)}{r+1}\right)_{r>8}$ is bounded above by 1.5. Since the three constituent sequences $\frac{\rho(r)}{3\,(r+1)}$, $\frac{(3\,r+2)r}{36\,\rho(r)^2}$ and $\frac{2\,r+1}{3\,\rho(r)}$ are bounded above, it follows that $C(r)$ is also bounded above. Being monotone increasing and bounded above $C(r)$ is convergent. Below we determine its limit.

Claim.

$$\lim_{r\to\infty} C(r) = \left(\frac{2 \times (13\sqrt{13} + 46)}{27}\right)^{1/3}. \tag{49}$$

Proof. Since

$$\rho(r) = \left(\frac{7}{6}\, r^3 + 2\, r^2 + \frac{1}{6}\, \sqrt{13\, r^2 + 8\, r + 1}\big(2\, r^2 + 3\, r + 1\big) + r + \frac{1}{6}\right)^{1/3},$$

we have

$$\lim_{r\to\infty} \frac{\rho(r)}{r} = \left(\frac{2}{6}\sqrt{13} + \frac{7}{6}\right)^{\frac{1}{3}} \tag{50}$$

Now,

$$C(r) = 2\sqrt{\frac{\rho(r)/r}{3\,(1+1/r)} + \frac{(3+2/r)}{36\,\rho(r)^2/r^2}} + \frac{2+1/r}{3\,\rho(r)/r} \tag{51}$$

Hence,

$$\begin{aligned}
\lim_{r\to\infty} C(r) &= 2\sqrt{\frac{(2\sqrt{13}+7)^{1/3}}{3\times 6^{1/3}} + \frac{3\times 6^{2/3}}{36\,(2\sqrt{13}+7)^{2/3}}} + \frac{2\times 6^{1/3}}{3\,(2\sqrt{13}+7)^{1/3}} \\
&= 2\sqrt{\frac{36(2\sqrt{13}+7)+54}{3\times 6^{1/3}\times 36\times (2\sqrt{13}+7)^{2/3}}} + \frac{2\times 6^{1/3}}{3\,(2\sqrt{13}+7)^{1/3}} \\
&= 2\sqrt{\frac{2(2\sqrt{13}+7)+3}{3\times 6^{1/3}\times 2\times (2\sqrt{13}+7)^{2/3}}} + \frac{2\times 6^{1/3}}{3\,(2\sqrt{13}+7)^{1/3}} \\
&= 2\sqrt{\frac{4\sqrt{13}+17}{3\times 6^{1/3}\times 2\times (2\sqrt{13}+7)^{2/3}}} + \frac{2\times 6^{1/3}}{3\,(2\sqrt{13}+7)^{1/3}} \\
&= 2\sqrt{\frac{(\sqrt{13}+2)^2}{3\times 6^{1/3}\times 2\times (2\sqrt{13}+7)^{2/3}}} + \frac{2\times 6^{1/3}}{3\,(2\sqrt{13}+7)^{1/3}} \\
&= \frac{2(\sqrt{13}+2)}{6^{2/3}\times (2\sqrt{13}+7)^{1/3}} + \frac{2\times 6^{1/3}}{3\,(2\sqrt{13}+7)^{1/3}} \\
&= \frac{6^{1/3}}{3(2\sqrt{13}+7)^{1/3}}\left[(\sqrt{13}+2)+2\right]
\end{aligned}$$

Thus,

$$\begin{aligned}
\lim_{r\to\infty} C(r) &= \frac{6^{1/3}(\sqrt{13}+4)}{3(2\sqrt{13}+7)^{1/3}} = \frac{6^{1/3}(61\sqrt{13}+220)^{1/3}}{3(2\sqrt{13}+7)^{1/3}} \\
&= \frac{6^{1/3}}{3}\left(\frac{61\sqrt{13}+220}{2\sqrt{13}+7}\right)^{1/3} \\
&= \frac{6^{1/3}}{3}\left(\frac{61\sqrt{13}+220}{2\sqrt{13}+7}\times\frac{2\sqrt{13}-7}{2\sqrt{13}-7}\right)^{1/3} \\
&= \frac{6^{1/3}}{3}\left(\frac{13\sqrt{13}+46}{3}\right)^{1/3} = \left(\frac{6\times(13\sqrt{13}+46)}{3\times 27}\right)^{1/3} \\
&= \left(\frac{2\times(13\sqrt{13}+46)}{27}\right)^{1/3}
\end{aligned}$$

□

The limit of $C(r)$ as r goes to infinity is the value of θ_1 where $L_Q(1/3, \theta_1)$ is the complexity of MNFS-GJL as determined in [21]. This shows that as r goes to infinity, the complexity of MNFS-New approaches the complexity of MNFS-GJL from below.

We have already seen that $C(r)$ is monotone increasing for $r \geq 1$. So, the minimum value of $C(r)$ is obtained for $r = 1$. After simplifying $C(1)$, we get the

minimum complexity of MNFS-New to be

$$L_Q\left(1/3, \left(\frac{146}{261}\sqrt{22}+\frac{208}{87}\right)^{1/3}\right) = L\left(1/3, 1.7116\right). \tag{52}$$

This minimum complexity is obtained at $c_p = \rho(1) = \left(\sqrt{22}+\frac{13}{3}\right)^{1/3} = 2.0819$.

Note 1. As mentioned earlier, for $r = k = 1$, the new method of polynomial selection becomes the Conjugation method. So, the minimum complexity of MNFS-New is the same as the minimum complexity for MNFS-Conj. Here we note that the value of the minimum complexity given by (52), is not same as the one reported by Pierrot in [21]. This is due to an error in the calculation in [21][3].

Medimum and large characteristic cases. In a manner similar to that used to prove Theorem 4, it is possible to work out the complexities for the medium and large characteristic cases of the MNFS corresponding to the new method. We have done this and the complexities turn out to be the same as that obtained in DBLP:conf/eurocrypt/Pierrot15 for the MNFS-GJL (for large characteristic) and the MNFS-Conjugation (for medium characteristic) methods. Hence, we do not present these details.

9 Conclusion

In this work, we have proposed a new method for polynomial selection for the NFS algorithm for fields $\mathbb{F}_{p^n}$ with $n > 1$. The complexity of the resulting NFS algorithm is parameterised by divisors d of n. For $d = 1$ or $d = n$, the obtained complexities are those of the GJL algorithm and the conjugate method proposed in [4]. For $1 < d < n$, the new method provides new trade-offs not obtained from earlier methods. The polynomial selection methods of [4] have been applied to the multiple number field sieve algorithm in [21]. In a similar manner, the new polynomial selection method proposed in this work has been applied to the multiple number field sieve algorithm. This leads to new state of the art complexity for the (M)NFS applied to the boundary case.

References

1. Leonard M. Adleman. The function field sieve. In Leonard M. Adleman and Ming-Deh A. Huang, editors, *ANTS*, volume 877 of *Lecture Notes in Computer Science*, pages 108–121. Springer, 1994.

[3] The error is the following. The solution for c_b to the quadratic $(18t^2c_p^2)c_b^2 - (36tc_p)c_b + 8 - 3t^2(t-1)c_p^3 = 0$ is $c_b = 1/(tc_p) + \sqrt{5/(9(c_pt)^2) + (c_p(t-1))/6}$ with the positive sign of the radical. In [21], the solution is erroneously taken to be $1/(tc_p) + \sqrt{5/((9c_pt)^2) + (c_p(t-1))/6}$

2. Leonard M. Adleman and Ming-Deh A. Huang. Function field sieve method for discrete logarithms over finite fields. *Inf. Comput.*, 151(1-2):5–16, 1999.
3. Shi Bai, Cyril Bouvier, Alain Filbois, Pierrick Gaudry, Laurent Imbert, Alexander Kruppa, François Morain, Emmanuel Thomé, and Paul Zimmermann. CADO-NFS, an implementation of the number field sieve algorithm. CADO-NFS, Release 2.1.1, 2014. `http://cado-nfs.gforge.inria.fr/`.
4. Razvan Barbulescu, Pierrick Gaudry, Aurore Guillevic, and Franois Morain. Improving nfs for the discrete logarithm problem in non-prime finite fields. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015*, volume 9056 of *Lecture Notes in Computer Science*, pages 129–155. Springer Berlin Heidelberg, 2015.
5. Razvan Barbulescu, Pierrick Gaudry, Antoine Joux, and Emmanuel Thom. A heuristic quasi-polynomial algorithm for discrete logarithm in finite fields of small characteristic. In PhongQ. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology EUROCRYPT 2014*, volume 8441 of *Lecture Notes in Computer Science*, pages 1–16. Springer Berlin Heidelberg, 2014.
6. Razvan Barbulescu and Cécile Pierrot. The multiple number field sieve for medium and high characteristic finite fields. *LMS Journal of Computation and Mathematics*, 17:230–246, 2014.
7. Daniel M. Gordon. Discrete logarithms in GF(p) using the number field sieve. *SIAM J. Discrete Math*, 6:124–138, 1993.
8. Robert Granger, Thorsten Kleinjung, and Jens Zumbrägel. Discrete logarithms in $gf(2^{9234})$. *NMBRTHRY list*, January 2014.
9. Aurore Guillevic. Computing individual discrete logarithms faster in GF(p^n). Cryptology ePrint Archive, Report 2015/513, 2015. `http://eprint.iacr.org/`.
10. Antoine Joux. Faster index calculus for the medium prime case application to 1175-bit and 1425-bit finite fields. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT*, volume 7881 of *Lecture Notes in Computer Science*, pages 177–193. Springer, 2013.
11. Antoine Joux. A new index calculus algorithm with complexity $L(1/4 + o(1))$ in small characteristic. In Tanja Lange, Kristin E. Lauter, and Petr Lisonek, editors, *Selected Areas in Cryptography - SAC 2013 - 20th International Conference, Burnaby, BC, Canada, August 14-16, 2013, Revised Selected Papers*, volume 8282 of *Lecture Notes in Computer Science*, pages 355–379. Springer, 2013.
12. Antoine Joux and Reynald Lercier. The function field sieve is quite special. In Claus Fieker and David R. Kohel, editors, *ANTS*, volume 2369 of *Lecture Notes in Computer Science*, pages 431–445. Springer, 2002.
13. Antoine Joux and Reynald Lercier. Improvements to the general number field sieve for discrete logarithms in prime fields. A comparison with the gaussian integer method. *Math. Comput.*, 72(242):953–967, 2003.
14. Antoine Joux and Reynald Lercier. Improvements to the general number field sieve for discrete logarithms in prime fields. A comparison with the Gaussian integer method. *Math. Comp.*, 72(242):953–967 (electronic), 2003.
15. Antoine Joux and Reynald Lercier. The function field sieve in the medium prime case. In Serge Vaudenay, editor, *EUROCRYPT*, volume 4004 of *Lecture Notes in Computer Science*, pages 254–270. Springer, 2006.
16. Antoine Joux, Reynald Lercier, Nigel Smart, and Frederik Vercauteren. The number field sieve in the medium prime case. In Cynthia Dwork, editor, *Advances in Cryptology - CRYPTO 2006*, volume 4117 of *Lecture Notes in Computer Science*, pages 326–344. Springer Berlin Heidelberg, 2006.

17. Antoine Joux, Reynald Lercier, Nigel P. Smart, and Frederik Vercauteren. The number field sieve in the medium prime case. In Cynthia Dwork, editor, *Advances in Cryptology - CRYPTO 2006, 26th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2006, Proceedings*, volume 4117 of *Lecture Notes in Computer Science*, pages 326–344. Springer, 2006.
18. Antoine Joux and Cécile Pierrot. The special number field sieve in $\mathbb{F}_{p^n}$ - Application to pairing-friendly constructions. In Zhenfu Cao and Fangguo Zhang, editors, *Pairing-Based Cryptography - Pairing 2013 - 6th International Conference, Beijing, China, November 22-24, 2013, Revised Selected Papers*, volume 8365 of *Lecture Notes in Computer Science*, pages 45–61. Springer, 2013.
19. M. Kalkbrener. An upper bound on the number of monomials in determinants of sparse matrices with symbolic entries. *Mathematica Pannonica*, 8(1):73–82, 1997.
20. D. Matyukhin. Effective version of the number field sieve for discrete logarithm in a field GF(p^k). *Trudy po Discretnoi Matematike 9, 121151 (2006) (in Russian)*, 2006. `http://m.mathnet.ru/php/archive.phtml?wshow=paper&jrnid=tdm&paperid=144&option_lang=eng`.
21. Cécile Pierrot. The multiple number field sieve with conjugation and generalized Joux-Lercier methods. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I*, volume 9056 of *Lecture Notes in Computer Science*, pages 156–170. Springer, 2015.
22. Palash Sarkar and Shashank Singh. Fine tuning the function field sieve algorithm for the medium prime case. Cryptology ePrint Archive, Report 2014/065, 2014. `http://eprint.iacr.org/`.
23. Oliver Schirokauer. Discrete logarithms and local units. *Philosophical Transactions: Physical Sciences and Engineering*, 1993.
24. Oliver Schirokauer. Using number fields to compute logarithms in finite fields. *Math. Comp.*, 69(231):1267–1283, 2000.
25. Oliver Schirokauer. Virtual logarithms. *J. Algorithms*, 57(2):140–147, Nov 2005.