Computing sparse multiples of polynomials

by

Hrushikesh Tilak

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2010

Abstract

We consider the problem of finding a sparse multiple of a polynomial. Given a polynomial $f \in \mathsf{F}[x]$ of degree d over a field F, and a desired sparsity $t = O(1)$, our goal is to determine if there exists a multiple $h \in \mathsf{F}[x]$ of f such that h has at most t non-zero terms, and if so, to find such an h.

When $\mathsf{F} = \mathbb{Q}$, we give a polynomial-time algorithm in d and the size of coefficients in h. For finding binomial multiples we prove a polynomial bound on the degree of the least degree binomial multiple independent of coefficient size.

When F is a finite field, we show that the problem is at least as hard as determining the multiplicative order of elements in an extension field of F (a problem thought to have complexity similar to that of factoring integers), and this lower bound is tight when $t = 2$.

Acknowledgements

I would like to thank my supervisor Prof. Mark Giesbrecht for his continuous encouragement over the past two years. His expertise and ideas helped me learn new techniques. I also thank him for guiding me while giving me sufficient independence to explore other problems and areas.

I also wish to thank my co-authors Mark Giesbrecht and Daniel S. Roche for many interesting discussions during the collaboration for [11]. Without their help, it would have been very difficult if not impossible to overcome several technical obstacles faced in the course of this work.

I am grateful to Prof. John May and Prof. Arne Storjohann for agreeing to read my thesis and providing numerous insightful comments.

Contents

Chapter 1

Introduction

Let F be a field, which will later be specified either to be the rational numbers ($\mathbb{Q}$) or a finite field with q elements ($\mathbb{F}_q$). We say a polynomial $h \in \mathsf{F}[x]$ is *t-sparse* (or *has sparsity t*) if it has at most t nonzero coefficients in the standard power basis; that is, h can be written in the form

$$h = h_1x^{e_1} + h_2x^{e_2} + \ldots + h_tx^{e_t} \quad \text{for } h_1, \ldots, h_t \in \mathsf{F} \text{ and } e_1, \ldots, e_t \in \mathbb{N}. \tag{1.1}$$

Sparse polynomials have a compact representation as a sequence of coefficient-degree pairs $(h_1, e_1), \ldots, (h_t, e_t)$, which allow representation and manipulation of very high degree polynomials. Let $f \in \mathsf{F}[x]$ have degree d. We examine the computation a t-sparse multiple of f. That is, we wish to determine if there exist $g, h \in \mathsf{F}[x]$ such that $fg = h$ and h has prescribed sparsity t, and if so, to find such an h. We do not attempt to find g, as it may have a super-polynomial number of terms, even though h has a compact representation (see Theorem 5.7).

Sparse multiples over finite fields have cryptographic applications, in particular correlation attacks on LFSR-based stream ciphers [7, 5]. Sparse multiples can facilitate efficient arithmetic in extension fields [3]. One of our original motivations was to understand the complexity of sparse polynomial implicitization: given a function generating zeros, find a sparse polynomial with those zeros (see [8]). The linear algebra formulation in Chapter 2 relates to to the finding the minimum distance of a binary linear code [2, 24] as well as finding "sparsifications" of linear systems [6].

In general, we assume that the desired sparsity t is a constant. This seems reasonable given that over a finite field, even for $t = 2$, the problem is probably computationally hard (Theorem 7.1). In fact, we have reason to conjecture that the problem is intractable over $\mathbb{Q}$ or $\mathbb{F}_q$ when t is a parameter. Our algorithms are singly exponential in t and polynomial in the other input parameters.

Over $\mathbb{Q}[x]$, the analysis must consider coefficient size, and we will count machine word operations in our algorithms to account for coefficient growth. We follow the conventions of [17] and define the *height* of a polynomial as follows. Let $f \in \mathbb{Q}[x]$ and $r \in \mathbb{Q}$ the least positive rational number such that $rf \in \mathbb{Z}[x]$. If $rf = \sum_i a_i x^{e_i}$ with each $a_i \in \mathbb{Z}$, then the *height* of f, written $\mathcal{H}(f)$, is $\max_i |a_i|$. It can be easily seen that scaling the input polynomial does not change this notion of size. This is very desirable for the problem at hand because the notion of existence of a sparse multiple is not changed if the input polynomial is multiplied by a scalar.

We examine variants of the sparse multiple problem over $\mathbb{F}_q$ and $\mathbb{Q}$. Since every polynomial f in $\mathbb{F}_q[x]$ has a 2-sparse multiple of high degree (namely $x^{q^e-1} - 1$ such that $\mathbb{F}_{q^e}$ is the splitting field of f), given $f \in \mathbb{F}_q[x]$ and $n \in \mathbb{N}$ we consider the problem of finding a t-sparse multiple of f with degree at most n. For input $f \in \mathbb{Q}[x]$ of degree d, we consider algorithms which seek t-sparse multiples of height bounded above by an additional input value $c \in \mathbb{N}$. We present algorithms requiring time polynomial in d and $\log c$.

The remainder of the thesis is structured as follows.

In Chapter 2, we consider the straightforward linear algebra formulation of the sparse multiple problem. This is useful over $\mathbb{Q}[x]$ once a bound on the output degree is derived, and also allows us to bound the output size.

In Chapter 3, we give an algorithm for finding the shortest ℓ_∞ vector in an integer lattice. This algorithm forms the basis for algorithms for finding sparse multiples of rationals multiples presented in the latter sections.

The linear algebra formulation of Chapter 2 connects our problem with related NP-complete coding theory problems. In Chapter 4, we present these problems after introducing basic notions relating to linear error-correcting codes.

In Chapter 5 we consider the problem of finding the least-degree binomial multiple of a rational polynomial. A polynomial-time algorithm in the size of the input is given which completely resolves the question in this case. This works despite the fact that we show polynomials with binomial multiples whose degrees and heights are both exponential in the input size!

In Chapter 6 we consider the more general problem of finding a t-sparse multiple of an input $f \in \mathbb{Q}[x]$. Given a height bound $c \in \mathbb{N}$ we present an algorithm which requires polynomial time in $\deg f$ and $\log c$, except when f has repeated cyclotomic factors.

Chapter 7 shows that, even for $t = 2$, finding a t-sparse multiple of a polynomial $f \in \mathbb{F}_q[x]$ is at least as hard as finding multiplicative orders in an extension of $\mathbb{F}_q$ (a problem thought to be computationally difficult). This lower bound is shown to be tight for $t = 2$ due to an algorithm for finding binomial multiples that uses order finding. Related problems considered in cryptography are discussed at the end of the chapter.

Open questions and avenues for future research are discussed in Chapter 8.

Most of the results in this thesis arose from work that was done in collaboration with Mark Giesbrecht and Daniel S. Roche. A conference version [11] will be presented at *International Symposium on Algorithms and Computation 2010*.

Chapter 2

Linear algebra formulation

The sparsest multiple problem can be formulated using linear algebra. This requires specifying bounds on degree, height and sparsity; later some of these parameters will be otherwise determined. This approach also highlights the connection to some problems from coding theory. We exhibit a randomized algorithm for finding a t-sparse multiple h of a rational polynomial f of degree d, given bounds c and n on the height and degree of the multiple respectively. The algorithm runs in time $(\log \mathcal{H}(f))^{O(1)} \cdot n^{O(t)}$ and returns the desired output with high probability.

Let R be a principal ideal domain, with $f \in \mathsf{R}[x]$ of degree d and $n \in \mathbb{N}$ given. Suppose $g, h \in \mathsf{R}[x]$ have degrees $n-d$ and n respectively, with $f = \sum_0^d f_i x^i$, $g = \sum_0^{n-d} g_i x^i$ and $h = \sum_0^n h_i x^i$. The coefficients in equation $fg = h$ satisfy the linear system (2.1).

$$\underbrace{\begin{bmatrix} f_0 & & & \\ f_1 & f_0 & & \\ \vdots & f_1 & \ddots & \\ f_d & \vdots & \ddots & f_0 \\ & f_d & \ddots & f_1 \\ & & \ddots & \vdots \\ & & & f_d \end{bmatrix}}_{A_{f,n}} \underbrace{\begin{bmatrix} g_0 \\ g_1 \\ \\ \vdots \\ \\ g_{n-d} \end{bmatrix}}_{v_g} = \underbrace{\begin{bmatrix} h_0 \\ h_1 \\ \\ \vdots \\ \\ \\ h_n \end{bmatrix}}_{v_h} \tag{2.1}$$

Thus, a multiple of f of degree at most n and sparsity at most t corresponds to a vector with at most t nonzero entries (i.e., a t-sparse vector) in the linear span of $A_{f,n}$.

If $f \in \mathsf{R}[x]$ is squarefree and has roots $\{\alpha_1, \cdots, \alpha_d\}$, possibly over a finite extension of R, then (2.2) also holds. Thus t-sparse multiples of a squarefree f correspond to t-sparse R-vectors in the nullspace of $A_n(\alpha_1, \ldots, \alpha_d)$:

$$\underbrace{\begin{bmatrix} 1 & \alpha_1 & \cdots & \alpha_1^n \\ 1 & \alpha_2 & \cdots & \alpha_2^n \\ \vdots & \vdots & \vdots & \vdots \\ 1 & \alpha_d & \cdots & \alpha_d^n \end{bmatrix}}_{A_n(\alpha_1,\ldots,\alpha_d)} \begin{bmatrix} h_0 \\ h_1 \\ \\ \vdots \\ \\ h_n \end{bmatrix} = \mathbf{0} \tag{2.2}$$

Formulation (2.2) helps in reasoning about sparse multiples when working over finite fields, and is used in Section 7.1.

Working over the rationals, an algorithm for finding sparse multiples of bounded height using the formulation (2.1) is presented in the following section.

2.1 Finding a sparse multiple of bounded height and degree

We now present an algorithm to find the sparsest bounded-degree, bounded-height multiple $h \in \mathbb{Q}[x]$ of an input $f \in \mathbb{Q}[x]$. Since height $\mathcal{H}$ of a polynomial is invariant under scaling, we may assume that $f, g, h \in \mathbb{Z}[x]$.

The basic idea is the following. Having fixed the positions at which the multiple h has nonzero coefficients, finding a low-height multiple is reduced to finding the nonzero vector with smallest ℓ_∞ norm in the image of a related matrix.

Let $I = \{i_1, \ldots, i_t\}$ be a t-subset of $\{0, \ldots, n\}$, and $A^I_{f,n} \in \mathbb{Z}^{(n+1-t)\times(n-d+1)}$ the matrix $A_{f,n}$ with rows $i_1, \ldots, i_t$ removed. Denote by $B^I_{f,n} \in \mathbb{Z}^{t\times(n-d+1)}$ the matrix consisting of the removed rows $i_1, \ldots, i_t$ of the matrix $A_{f,n}$.

Existence of a t-sparse multiple $h = h_{i_1}x^{i_1} + h_{i_2}x^{i_2} + \cdots + h_{i_t}x^{i_t}$ of input f is equivalent to the existence of a vector v_g such that $A^I_{f,n} \cdot v_g = \mathbf{0}$ and $B^I_{f,n} \cdot v_g = [h_{i_1}, \ldots, h_{i_t}]^T$.

As an example, consider the instance where $d = 4$, $n = 7$ and the 3-sparse multiple is $h_0 + h_3x^3 + h_7x^7$ (the grey rows correspond to the set I):

$$\begin{bmatrix} f_0 & 0 & 0 & 0 \\ f_1 & f_0 & 0 & 0 \\ f_2 & f_1 & f_0 & 0 \\ f_3 & f_2 & f_1 & f_0 \\ f_4 & f_3 & f_2 & f_1 \\ 0 & f_4 & f_3 & f_2 \\ 0 & 0 & f_4 & f_3 \\ 0 & 0 & 0 & f_4 \end{bmatrix} \begin{bmatrix} g_0 \\ g_1 \\ g_2 \\ g_3 \end{bmatrix} = \begin{bmatrix} h_0 \\ 0 \\ 0 \\ h_3 \\ 0 \\ 0 \\ 0 \\ h_7 \end{bmatrix}$$

Now let $C^I_{f,n}$ be a matrix whose columns span the nullspace of the matrix $A^I_{f,n}$. Since $\mathsf{R}[x]$ is an integer domain, $A_{f,n}$ has no non-trivial null vector, and thus has full column rank. Since at most t rows are removed from $A_{f,n}$, the nullspace of $A^I_{f,n}$ has dimension $s \le t$, and hence $C^I_{f,n} \in \mathbb{Z}^{(n-d+1)\times s}$. Thus, a t-sparse multiple $h = h_{i_1}x^{i_1} + \cdots + h_{i_t}x^{i_t}$ of f exists if and only if there exists a $v \in \mathbb{Z}^s$ such that

$$B^I_{f,n} \cdot C^I_{f,n} \cdot v = [h_{i_1}, \ldots, h_{i_t}]^T. \tag{2.3}$$

Note that $B^I_{f,n} \cdot C^I_{f,n} \in \mathbb{Z}^{t\times s}$.

Algorithm 2.1 outlines this approach. The following lemma shows how to compute Step 5 efficiently using the Smith normal form.

Lemma 2.1. *Given $T \in \mathbb{Z}^{k\times\ell}$ with $k \ge \ell$ and nullspace of dimension s, we can compute a $V \in \mathbb{Z}^{\ell\times s}$ such that the image of V equals the nullspace of T. The algorithm requires $O\tilde{~}(k\ell^2 s \log \|T\|)$ bit operations (ignoring logarithmic factors, and $\|T\|$ denoting the largest magnitude of an entry in T).*

Proof. First compute the Smith normal form of the matrix: $T = PSQ$ for diagonal matrix $S = \operatorname{diag}(\delta_1, \ldots, \delta_{\ell-s}, 0, \ldots, 0) \in \mathbb{Z}^{k\times\ell}$ and unimodular matrices $P \in \mathbb{Z}^{k\times k}$ and $Q \in \mathbb{Z}^{\ell\times\ell}$. [23] gives efficient algorithms to compute such a P, S, Q with $O\tilde{~}(k\ell^2 s \log \|T\|)$ bit operations.

Algorithm 2.1: Bounded-Degree Bounded-Height Sparsest Multiple

```
Input: $f \in \mathbb{Z}[x]$ and $t, n, c \in \mathbb{N}$
Output: A $t$-sparse multiple $h$ of $f$ with $\deg(h) \le n$ and $\mathcal{H}(h) \le c$, or
        "NONE"
1 for $s = 2, 3, \ldots, t$ do
2     foreach $s$-subset $I = (1, i_2, \ldots, i_s)$ of $\{1, 2, \ldots, n\}$ do
3         Compute matrices $A^I_{f,n}$ and $B^I_{f,n}$ as defined above
4         if $A^I_{f,n}$ does not have full column rank then
5             Compute matrix $C^I_{f,n}$, a kernel basis for $A^I_{f,n}$
6             $\mathbf{h} \leftarrow$ shortest $\ell_\infty$ vector in the column lattice of $B^I_{f,n} \cdot C^I_{f,n}$
7             if $\ell_\infty(\mathbf{h}) \le c$ then return $h_1 + h_2 x^{i_2} + \cdots + h_t x^{i_t}$
8 return "NONE"
```

Then since any vector $\mathbf{v}$ in the nullspace of T satisfies $PSQ\mathbf{v} = \mathbf{0}$, $SQ\mathbf{v} = \mathbf{0}$ also and $\mathbf{v}$ is in the nullspace of SQ. Next compute the inverse of Q; this can be accomplished with the same number of bit operations since $\ell \le k$. Define V to be the last s columns of Q^{-1}.

$$\underbrace{\begin{bmatrix} \ddots & \\ & \mathbf{0}_s \end{bmatrix}}_{S} Q\mathbf{v} = \mathbf{0} \Leftrightarrow \exists \mathbf{u}\colon\ Q\mathbf{v} = \begin{bmatrix} \mathbf{0}_{(l-s)\times s} \\ \mathbf{I}_s \end{bmatrix} \mathbf{u} \Leftrightarrow \exists \mathbf{u}\colon\ \mathbf{v} = \underbrace{Q^{-1} \begin{bmatrix} \mathbf{0}_{(l-s)\times s} \\ \mathbf{I}_s \end{bmatrix}}_{V} \mathbf{u}$$

Due to the diagonal structure of S, V must be a nullspace basis for SQ, and furthermore V has integer entries since Q is unimodular. □

Lemma 2.2 shows that Step 6 can be performed efficiently.

Lemma 2.2. *The shortest integer vector, under the ℓ_∞ norm, in the image of a matrix $U \in \mathbb{Z}^{t\times t}$ can be computed by a randomized algorithm which performs $2^{O(t \log t)} t^{O(1)}$ arithmetic operations.*

Proof. Interpret the columns of the matrix U as the basis of a lattice $\mathcal{L} \subseteq \mathbb{R}^t$; integer vectors in the image of U correspond to vectors in the lattice $\mathcal{L}$.

Denote by v_∞ the shortest nonzero vector in the lattice under the ℓ_∞ norm. We have that $\ell_2(v_\infty) \le \sqrt{t} \cdot \ell_\infty(v_\infty) \le \sqrt{t} \cdot \ell_\infty(v) \le \sqrt{t} \cdot \ell_2(v)$, for any nonzero vector $v \in \mathcal{L}$. The first inequality is true for any vector v in place of v_∞ because $\sqrt{\sum_i v_i^2} \leq \sqrt{\sum_i \ell_\infty(v)^2} \leq \sqrt{t} \cdot \ell_\infty(v)$. The second inequality follows from the definition of v_∞, the vector minimizing the ℓ_∞ norm. The third inequality follows from $\sum_i v_i^2 \geq \ell_\infty(v)^2$, and hence that $\ell_2(v) = \sqrt{\sum_i v_i^2} \geq \ell_\infty(v)$.

Hence, the ℓ_2 norm of the shortest ℓ_∞ vector is at most $\sqrt{t}$ times the ℓ_2 norm of the shortest ℓ_2 vector.

[1] give a randomized algorithm which, with high probability, returns the shortest ℓ_2 vector in an integer lattice. They observe that this algorithm can be modified to return all vectors v such that $\ell_2(v) \leq \nu \ell_2(v_2)$ for a constant ν and $v_2 \in \mathcal{L}$ the shortest ℓ_2 vector. Using this modified version with $\nu = \sqrt{t}$, the above observations guarantee that the shortest ℓ_∞ vector in $\mathcal{L}$ will be returned with high probability. Since this modification differs from the original in a fairly subtle manner, we find merit in working through the changes and addressing the technical subtleties in Chapter 3.

□

The correctness and efficiency of Algorithm 2.1 can be summarized as follows.

Theorem 2.3. *Algorithm 2.1 correctly computes a t-sparse multiple h of f of degree n and height c, if it exists, with $(\log \mathcal{H}(f))^{O(1)} \cdot n^{O(t)}$ bit operations. The sparsity s of h is minimal over all multiples with degree less than n and height less than c, and the $\deg h$ is minimal over all such s-sparse multiples.*

Proof. The total number of iterations of the for loops is $\sum_{s=2}^{t} \binom{n-1}{s-1} < n^t$. Computing the rank of $A^I_{f,n}$, and computing the matrices $B^I_{f,n}$ and $C^I_{f,n}$ can each be done in polynomial time by Lemma 2.1. The size of the entries of $C^I_{f,n}$ is bounded by some polynomial $(\log \mathcal{H}(h) + n)^{O(1)}$. The computation of the shortest ℓ_∞ vector can be done using 2^{ut} operations on numbers of length $(\log \mathcal{H}(h) + n)^{O(1)}$ where the constant u depends only on t, by Lemma 2.2.

The minimality of sparsity and degree comes from the ordering of the for loops. Specifically, the selection of subsets in Step 2 is performed in *reverse lexicographic order*, so that column subsets I corresponding to lower degrees are always searched first. □

In this section, (using the algorithm from the following section) we have presented a randomized algorithm for finding bounded height t-sparse multiples of rational polynomials. For $t = 2$, we give a deterministic algorithm without assuming a priori height bounds in Section 5.

Chapter 3

Finding short ℓ_∞ vectors in lattices

In this chapter, we present an algorithm to find the shortest ℓ_∞ vector in a lattice. As mentioned earlier, this is a modification of the algorithm presented in [1].[†]

Algorithm 3.1 finds the shortest ℓ_∞ vector in a lattice whose shortest ℓ_2 vector has length λ_1 satisfying $2 \leq \lambda_1 < 3$. This algorithm can be made to work for any lattice by scaling. More precisely, given a lattice $\mathcal{L}$, we first run the [16] algorithm to get an approximation λ for $\lambda_1(\mathcal{L})$: $\lambda_1(\mathcal{L}) \leq \lambda \leq 2^n\lambda_1(\mathcal{L})$. Let v_k be the vector returned by Algorithm 3.1 when given as input the lattice $\frac{1.5^k}{\lambda}\mathcal{L}$ (that is, the basis vectors of $\mathcal{L}$ multiplied by $\frac{1.5^k}{\lambda}$) for each $k \in \{1, 2, 3, \ldots, 2n\}$. For some k in this range, $2 \leq \lambda_1(\frac{1.5^k}{\lambda}\mathcal{L}) < 3$ holds. For such a k, v_k is the shortest ℓ_∞ vector in the corresponding lattice, and hence $\frac{\lambda}{1.5^k}v_k$ is the shortest ℓ_∞ vector in the given lattice. Thus it suffices to output the shortest ℓ_∞ vector among $\{\frac{\lambda}{1.5^k}v_k\}$.

We will now prove the correctness of Algorithm 3.1 when presented with a lattice $\mathcal{L}$ such that $2 \leq \lambda_1(L) < 3$.

[†]Our modified version of the algorithm is based on the presentation of [1] by Oded Regev and his students. The lecture notes can be found at the course webpage: `http://www.cs.tau.ac.il/~odedr/teaching/lattices_fall_2004/`

Algorithm 3.1: Shortest ℓ_∞ vector in a lattice

```
Input: Basis $B = \{b_1, \dots, b_n\}$ for lattice $\mathcal{L} \subseteq \mathbb{Z}^n$ such that
       $2 \leq \lambda_1(\mathcal{L}) < 3$
Output: Shortest $\ell_\infty$ vector in $\mathcal{L}$
1  foreach $\gamma \in \{(3/2)^2, (3/2)^3, \dots\} \cap [0, 3\sqrt{n} + 1]$ do
2      $R_0 \leftarrow n \max_i \|b_i\|$
3      $N \leftarrow \lceil 2^{(7+\lceil\log(\gamma)\rceil)n} \log(R_0) \rceil$
4      Sample points $\{x_1, \dots, x_N\}$ uniformly and independently from
       $\mathbf{B}_n(0, \gamma)$, the $n$-dimensional ball of radius $\gamma$ centered around $\mathbf{0}$
5      $Z \leftarrow \{1, 2, \dots, N\}$
6      $y_i \leftarrow x_i \mod \mathcal{P}(B)$ for every $i \in Z$; $\mathcal{P}(B)$ being the parallelogram
       of $B$, defined in proof of 3.2
7      $R \leftarrow R_0$
8      while $R > 2\gamma + 1$ do
9          $J \leftarrow \emptyset$
10         foreach $i \in Z$ do
11             if $min_{j \in J} \|y_j - y_i\| > R/2$ then
12                 $J \leftarrow J \cup \{i\}$
13         $Z \leftarrow Z \setminus J$
14         $y_i \leftarrow y_i + x_{\eta(i)} - y_{\eta(i)}$ for $i \in Z$
15         $R \leftarrow R/2 + \gamma$
16     $Y_\gamma \leftarrow \{(x_i - y_i) | i \in Z\}$
17 $v_\infty \leftarrow \arg\max_{v-w} \|v - w\|_\infty$ for $v, w \in Y_\gamma$ and $v \neq w$, and all $\gamma$
18 return $v_\infty$
```

As mentioned in the proof of Lemma 2.2, to find the shortest ℓ_∞ vector v_∞ in a lattice, it suffices to consider all lattice vectors of ℓ_2 norm at most $\sqrt{n}$ times the norm of the shortest ℓ_2 vector. Algorithm 3.1 achieves this by running the main body of the loop with different values of γ. In a particular iteration of the outermost loop, with high probability, the algorithm encounters all lattice vectors v with ℓ_2 norm $d = \|v\|$ such that $2d/3 \leq \gamma < d$. Call all such v *interesting.* By iterating over a suitable range of γ, it encounters (with high probability) all *interesting* vectors. Finally, it returns the shortest ℓ_∞ vector among them, which with high probability is the shortest ℓ_∞ vector in the lattice.

For a particular iteration of the loop (with a fixed γ), the algorithm uniformly[†] samples a large number of vectors from an appropriately sized ball. It then performs a series of *sieving* steps to ensure that at the end of these steps, it is left with lattice vectors of sufficiently small ℓ_2 norm. Using a probabilistic argument, it is argued that all *interesting* vectors are obtained.

The following lemma proves the correctness of the *sieving* steps. These correspond to steps 9 to 12 of the algorithm. At the end of this *sieving* the algorithm produces a J of size at most 5^n.

Lemma 3.1. *Given $\{y_1, \ldots, y_N\} \subseteq \mathbb{R}^n$ such that $\|y_i\| \leq R$, the following can be efficiently computed: a subset $J \subseteq [N]$ of size at most 5^n and a mapping $\eta : [N] \to J$ such that $\left\|y_i - y_{\eta(i)}\right\| \leq R/2$.*

Proof. Initially the set J is set to be empty. The algorithm iterates over the points y_i. It adds i to J only if $min_{j \in J}(\|y_j - y_i\|) > R/2$. It sets $\eta(j) = j$ for $j \in J$. For $i \notin J$, it sets $\eta(i)$ to a $j \in J$ such that $\|y_j - y_i\| \leq R/2$.

It is clear that this procedure runs in polynomial time. To see that the size of J is at most 5^n, note that all the balls of radius $R/4$ and centered at y_j for $j \in J$ are disjoint. This is because every pair of points in J are separated by a distance greater than $R/2$, by construction of J. Also, these balls are contained in a ball of radius $R + R/4$ since $\|y_i\| \leq R$. Thus the total number of disjoint balls, and hence the size of J, can be bounded above by comparing the volumes: $|J| \leq (\frac{5R/4}{R/4})^n = 5^n$ $\square$

[†]The analysis and the proof of correctness of the algorithm work even if an almost-uniform sampling over rational vectors of sufficiently large description is performed. This is because the description of sufficiently small lattice vectors is only a polynomial in size of the basis vectors. This obviates precision and representation issue concerns.

The algorithm views every sampled vector x_i as a perturbation of a lattice vector $x_i - y_i$ for some y_i. The idea is the following: initially y_i is calculated so that x_i is a perturbation of some large lattice vector. Iteratively, the algorithm either obtains shorter and shorter lattice vectors corresponding to x_i, or discards x_i in some sieving step. At all stages of the algorithm, $x_i - y_i$ is a lattice vector. These observations are made concrete by the following two lemma.

Lemma 3.2. *$\{y_i\}$ can be found efficiently in Step 6; and $\{x_i - y_i\} \subseteq \mathcal{L}$.*

Proof. For a fixed x_i, y_i is set to $(x_i \mod \mathcal{P}(B))$ where $\mathcal{P}(B)$ denotes all vectors contained in the parallelogram $\{\sum_{i=1}^{n} \alpha_i b_i | 0 \leq \alpha_i < 1\}$, with b_i being the given basis vectors. Thus y_i is the unique element in $\mathcal{P}(B)$ such that $y_i = x_i - v$ for $v \in \mathcal{L}$. From this definition of y_i, we get that $x_i - y_i \in \mathcal{L}$ for every i.

It is easy to see that y_i can be calculated efficiently: simply represent y_i as a rational linear combination of the basis vectors $\{b_i\}$ and then truncate each coefficient modulo $[0, 1)$. □

Lemma 3.3. $Y_\gamma \subset \mathcal{L} \cap \mathbf{B}_n(0, 3\gamma + 1)$.

Proof. By Lemma 3.2, $(x_i - y_i) \in \mathcal{L}$ for all i before the start of the loop. It needs to be proved that the same holds after the loop, and furthermore, all the resulting lattice vectors lie in $\mathbf{B}_n(0, 3\gamma + 1)$. Whenever the algorithm modifies any y_i, it sets it to $y_i + x_{\eta(i)} - y_{\eta(i)}$; and thus a lattice vector $(x_i - y_i)$ changes into $(x_i - y_i) - (x_{\eta(i)} - y_{\eta(i)})$. Since both of the terms are lattice vectors, so is their difference. Thus $Y_\gamma \subset \mathcal{L}$.

We will now show that the invariant $y_i \leq R$ is maintained at the end of every iteration. This suffices to prove that $x_i - y_i \in \mathbf{B}_n(0, 3\gamma + 1)$ because $x_i \in \mathbf{B}_n(0, \gamma)$ and $\|y_i\| \leq 2\gamma + 1$ by loop termination condition.

Initially, $y_i = \sum_{j=1}^{n} \alpha_j b_j$ for some coefficients α_j satisfying $0 \leq \alpha_j < 1$. Thus $\|y\| \leq \sum_j \|b_j\| \leq n \max_j b_j \leq R_0 = R$. Consider now the result of the change $y_i \to y_i + x_{\eta(i)} - y_{\eta(i)}$. We have that $\|y_i + x_{\eta(i)} + y_{\eta(i)}\| \leq \|y_i - y_{\eta(i)}\| + \|x_{\eta(i)}\|$. First of these terms is bounded by $R/2$ because of construction of $\eta(.)$ in Lemma 3.1. From $\|x_i\| \leq \gamma$, we get that $\|y_i\| \leq R/2 + \gamma$. Since the value of R gets updated appropriately, the invariant $\|y_i\| \leq R$ is maintained at the end of the loop. □

The following crucial lemma says that Y_γ can be used to compute all *interesting* vectors:

Lemma 3.4. *Let $v \in \mathcal{L}$ be a lattice vector such that $\frac{2}{3}\|v\| \leq \gamma < \|v\|$. Then, with probability at least $1 - 1/2^{O(n)}$, Y_γ contains both w and $w \pm v$ for some w.*

Using this lemma, we can prove our main theorem:

Theorem 3.5. *Algorithm 3.1 runs in time $2^{O(n \log n)} n^{O(1)}$ and outputs the shortest ℓ_∞ vector with probability at least $1 - 1/2^{O(n)}$.*

Proof. Since the length $\lambda_1(\mathcal{L})$ of the shortest ℓ_2 vector is assumed to satisfy $2 \leq \lambda_1(L) < 3$, we have that the ℓ_2 norm of the shortest ℓ_∞ vector v_∞ satisfies $\|v_\infty\| < 3\sqrt{n}$. Since the algorithm iterates over several γ in an appropriate range, by Lemma 3.4, some Y_γ contains w and $w \pm v_\infty$ for some w with high probability. Since the algorithm computes the differences of the vectors in Y_γ, it outputs v_∞ with high probability.

The number of sampled points in each iteration of the outer loop is $2^{O(n \log \gamma)}$. Since $\gamma = O(\sqrt{n})$ and each arithmetic operation takes time $n^{O(1)}$, the algorithm runs in time $2^{O(n \log n)} n^{O(1)}$. $\square$

To prove Lemma 3.4, a probabilistic argument will be employed. The proof can be broken into three steps. First, identify a set of *good* points from the sampled points, and argue that this set is large. Next, argue that there must exist a lattice point which corresponds to lots of *good* points. Finally, argue that an imaginary probabilistic step does not essentially change the behaviour of the algorithm. Combined with the existence of a lattice point corresponding to many *good* points, this imaginary step allows us to argue that the algorithm encounters both w and $w \pm v$ for an appropriate *interesting* v.

Let v be an *interesting* lattice vector. That is, $\frac{2}{3}d \leq \gamma < d$ for $d = \|v\|$. For the iteration where the algorithm uses a value of γ in this range, we will denote by C_1 the points in the set $\mathbf{B}_n(v, \gamma) \cap \mathbf{B}_n(0, \gamma)$. Similarly, $C_2 = \mathbf{B}_n(-v, \gamma) \cap \mathbf{B}_n(0, \gamma)$. By choice of γ, C_1 and C_2 are disjoint. We will call the points in $C_1 \cup C_2$ as **good**. Following lemma shows that probability of sampling a *good* point is large.

Lemma 3.6. $\mathbf{Pr}[x_i \in C_1] \geq 2^{-2n}$

Proof. The radius of both $\mathbf{B}_n(0, \gamma)$ and $\mathbf{B}_n(v, \gamma)$ is γ. The distance between the centers is $d = \|v\|$. Thus the intersection contains a sphere of radius $\frac{1}{2}(2\gamma - d) = \gamma - d/2$ whose volume $Vol(\mathbf{B}_n(0, \gamma - d/2))$ gives a lower bound on the volume of C_1. Comparing with $Vol(\mathbf{B}_n(0, \gamma))$ and using the fact that $\gamma \geq \frac{2}{3}d$, we get that $\mathbf{Pr}[x_i \in C_1] \geq \frac{Vol(\mathbf{B}_n(0,\gamma - d/2))}{Vol(\mathbf{B}_n(0,\gamma))} \geq \left(\frac{\gamma/4}{\gamma}\right)^n = 2^{-2n}$. □

Informally, the following lemma says that even if the size of Z is large at the end of the inner loop, the set $\{x_i - y_i\}$ has many repetitions.

Lemma 3.7. $|Y_\gamma| \leq (3\gamma + 2)^n$

Proof. The points in $\mathcal{L}$ are separated by a distance of at least 2 since $\lambda_1(\mathcal{L}) \geq 2$. Hence balls of radius 1 around each lattice point are pairwise disjoint. If we consider only the balls corresponding to points in Y_γ, all of them are contained in a ball of radius $3\gamma + 2$ since $Y_\gamma \subseteq \mathbf{B}_n(0, 3\gamma + 1)$ by Lemma 3.2. Thus the total number of points in Y_γ is at most $\frac{Vol(\mathbf{B}_n(0,3\gamma+2))}{Vol(\mathbf{B}_n(0,1))} = (3\gamma + 2)^n$. □

The following lemma argues that there must be a lattice point corresponding to many *good* points.

Lemma 3.8. *With high probability, there exists $w \in Y_\gamma$ and $I \subseteq Z$ such that $|I| \geq 2^{3n}$, and for all $i \in I$, $x_i \in C_1 \cup C_2$ and $w = x_i - y_i$.*

Proof. Since $\mathbf{Pr}[x_i \in C_1 \cup C_2]$ is at least 2^{-2n} by Lemma 3.6, and the number of points sampled is $\lceil 2^{(7+\lceil \log(\gamma) \rceil)n} \log(R_0) \rceil$, the expected number of *good* points sampled at the start is at least $2^{(5+\lceil \log(\gamma) \rceil)n} \log(R_0)$. The loop performs $\log(R_0)$ iterations removing (by Lemma 3.1) at most 5^n points per iteration. The total number of *good* points remaining in Z after the sieving steps is $(2^{(5+\lceil \log(\gamma) \rceil)n} - 5^n) \log(R_0) \geq 2^{(4+\lceil \log(\gamma) \rceil)n} \log(R_0)$ since $5^n \leq 2^{3n}$.

By Lemma 3.7, $|Y_\gamma| \leq (3\gamma + 2)^n$. Since $3\gamma + 2 \leq 4\gamma$ for $\gamma \geq (3/2)^2$, $|Y_\gamma| \leq 2^{(2+\log(\gamma))n}$. Hence, there exists a $w \in Y_\gamma$ corresponding to at least $\frac{2^{(4+\lceil \log(\gamma) \rceil)n} \log(R_0)}{2^{(2+\log(\gamma))n}} \geq 2^{3n}$ *good* points. □

The final step in the analysis is to argue that for such a $w \in Y_\gamma$, we must also have that $w \pm v \in Y_\gamma$ with high probability for an *interesting* $v \in \mathcal{L}$.

Proof of Lemma 3.4

Consider the iteration where γ satisfies $\frac{2}{3}\|v\| \leq \gamma < \|v\|$ for an *interesting* lattice vector v.

It can be easily seen that $x \in C_1$ if and only if $x - v \in C_2$. Consider an imaginary process performed just after sampling all the x_i. For each $x_i \in C_1$, with probability $1/2$, we replace it with $x - v \in C_2$. Similarly, for each $x \in C_2$, we replace it with $x + v \in C_1$. (This process cannot be performed realistically without knowing v, and is just an analysis tool.) The definition of y_i is invariant under addition of lattice vectors $v \in \mathcal{L}$ to x_i, and hence the y_i remain the same after this process.

Since the sampling was done from the uniform distribution and since $(x \in C_1) \leftrightarrow (x - v \in C_2)$ is a bijection, this process does not change the sampling distribution.

We may postpone the probabilistic transformation $x_i \leftrightarrow (x_i - v)$ to the time when it actually makes a difference. That is, just before the first time when x_i is used by the algorithm. The algorithm uses x_i in two places. For $i \in J$ during the *sieving* step, we perform this transformation immediately after computation of J. Another place where x_i is used is the computation of Y_γ. We perform this transformation just before this computation.

In the original algorithm (without the imaginary process), by Lemma 3.8, there exists a point $w \in Y_\gamma$ corresponding to at least 2^{3n} *good* points. Let $\{x_i\}$ be this large set of *good* points. With high probability, there will be many x_i which remain unchanged, and also many x_i which get transformed into $x_i \pm v$. Thus, Y_γ contains both w and $w \pm v$ with high probability. $\square$

Chapter 4

Coding theory problems

All our algorithms require time exponential in t, and are only polynomial-time for constant t. It is natural to ask whether there are efficient algorithms which require time polynomial in t. Unfortunately, we conjecture this problem is probably NP-complete. We give some evidence for the conjectured hardness by pointing out related problems from coding theory.

First we introduce some basic terminology from coding theory. Then we introduce some NP-hard problems, and display how they relate to the problems at hand.

4.1 Linear error-correcting codes

In this chapter we introduce some basic notions from coding theory which will be used in subsequent sections.

The basic premise of coding theory is to transmit messages across noisy channels by introducing structured redundancy. The simplest kind of protocol, called *Forward Error Correction* in the literature, begins with the sender encoding the message into a longer encoded message which is transmitted across a noisy channel. The receiver obtains a potentially corrupted version of the encoded message. Using a decoding algorithm, the original message can be obtained from the corrupted version of the encoding because of the carefully added redundancy.

Depending on the nature (randomized, adversarial, etc.) of the noise in the channel and the desired efficiency of the encoding and decoding algorithms, various kinds of protocols are chosen which promise successful transmission with high probability.

In the theory of error-correcting codes, the following characterizes many commonly seen protocols: The alphabet is a finite sized set, usually F_2 or other small finite fields. All messages are of length m, and all encodings are of length $n \geq m$. The noise is assumed to be adversarial; there exists a constant k ($0 \leq k \leq n$), and the adversary can modify any k positions of the encoded message.

Thus, an error-correcting code is a subset $C \subseteq \mathsf{F}^n$ of size $|F|^m$.

We will assume all of the above in this section. Furthermore, we will assume existence of two randomized algorithms $\mathcal{E}$ and $\mathcal{D}$, the encoding and decoding algorithms respectively. The encoding algorithm $\mathcal{E}$ takes as input a message string in F^m, and outputs a string in F^n. The decoding algorithm $\mathcal{D}$ takes as input a string from F^n, and outputs a string in F^m. Furthermore, it is guaranteed that for all corrupted versions of a codeword with at most a fixed number of corruptions, $\mathcal{D}$ outputs the original message with high probability. To make this rigorous, we need the notion of *distance* of an error-correcting code.

Definition 4.1. *Let $d : \mathsf{F}^n \times \mathsf{F}^n \to \mathbb{N}$ denote the Hamming distance function, which counts the number of indices in which two vectors differ. The* distance *of an error-correcting code $C \subseteq F^n$ is defined to be $\min_{x \neq y} d(x, y)$ over $x, y \in C$.*

The distance of an error-correcting code is a very important parameter which influences how many errors the code can tolerate. As an example, for the simplest decoding scheme of outputting the message with codeword nearest to the received word, it is easy to see that a code with distance d can tolerate up to $\lfloor (d-1)/2 \rfloor$ errors. That is to say, given that the codeword is corrupted in at most $\lfloor (d-1)/2 \rfloor$ positions, the receiver can correctly obtain the original message.

One well-studied family of encoding schemes is linear error-correcting codes:

Definition 4.2. *A* linear error-correcting code *$\mathcal{E}$ is a linear mapping $\mathcal{E} : \mathsf{F}^m \to F^n$ specified by a matrix $\mathbf{M} \in F^{m \times n}$.*

There are two matrices associated with a linear error-correcting code. The generator matrix $\mathbf{G} \in \mathsf{F}^{n \times m}$ defines the encoding algorithm: Interpret the message as a vector $v \in \mathsf{F}^m$, and then, $\mathbf{G}v$ defines the associated codeword. The second matrix $\mathbf{H} \in \mathsf{F}^{(n-m) \times m}$, called the parity-check matrix, assists in detecting if a received message is an encoded message: $\mathbf{H}u = 0 \iff \exists v\colon \mathbf{G}v = u$.

The popularity of this family arises from the fact that the encoding algorithms are usually very efficient. In the case where the message is transmitted without errors, very efficient decoding algorithms also exist. As we will see later, the problem of decoding with errors is NP-hard.

4.2 Problems

Maximum likelihood decoding

Under the presence of adversarial noise (and even in the case of random noise), one possible decoding strategy is to output the codeword which minimizes the number of errors. Under general probabilistic error models, this translates into the following: Given a string $y \in \mathsf{F}^n$, output the codeword $x \in F^n$ such that $\mathbf{Pr}[\text{y is received}|\text{x is sent}]$ is maximized. Under adversarial noise, this translates into finding the codeword nearest (under the Hamming metric) to the received message.

The problem of Maximum Likelihood Decoding is as follows:

Maximum Likelihood Decoding: Given a matrix $\mathbf{H} \in \mathsf{F}^{(n-m) \times n}$, a vector $s \in \mathsf{F}^{n-m}$ and an integer w, is there a vector $x \in \mathsf{F}^n$ such that $\mathbf{H}x = s$ and Hamming weight $|x|$ of x is less than w?

When viewed as a linear algebra problem, this translates into finding the sparsest solution to a system of linear equations. [2] proved that this problem is NP-hard over F_2. While this result shows NP-hardness for very general matrices, [13] showed that this problem NP-hard for the class of Reed-Solomon codes. These codes interpret the input as a degree $m-1$ polynomial. The encoding consists of evaluating these polynomials at n fixed points (using, for instance, the formulation in (2.2)).

[13] use a matrix similar to the one in Equation 2.2 in their hardness proof. However, this falls short of proving the hardness of finding sparse multiples for square-free polynomials because the maximum likelihood problem allows (and essentially utilizes in hardness proof) choice of the right hand side while in the case of the sparse multiple problem, it is zero. The natural question of whether similar techniques can be used to prove hardness for our problem remains unanswered.

Minimum distance problem

Our definitions above do not preclude very different algorithms for different message lengths. However, we will be mostly interested in "uniform" families of codes. One possible condition that can be imposed is that there is an encoding algorithm which works for messages of all lengths.

For such uniform families of codes, usually the goal is to optimize two parameters. The first parameter is the length of the encoding. The second parameter is distance of the error-correcting code.

Reducing the length of the encoding facilitates cheaper transmission costs. Increasing the distance ensures that the algorithm works under large amount of corruptions. It should be intuitively clear that these two goals are orthogonal. Shannon's theorem makes these notions rigorous, and gives a lower bound on the length of the encodings in terms of distance and other related parameters.

On the other hand Gilbert-Varshamov bound asserts that there exists error-correcting codes with sufficiently small encoding lengths. Attaining this bound is a primary goal in designing error-correcting codes. It is known that *random* linear error-correcting codes achieve this bound, and hence a protocol choosing a random encoding scheme for every length performs very well in practice.

An algorithm for computing the minimum distance of a linear error-correcting code would ensure generation of codes with guarantees on distance of the code. Formally, the problem can be defined as follows:

MD-Linear (*Minimum distance of a linear error-correcting code*)**:** Given an integer w and a linear error-correcting code $\mathcal{C}$, is there a codeword of Hamming weight at most w?

If $\mathcal{C}$ is specified by its generator matrix $\mathbf{G}$, the problem is to find a sparse vector in the image of $\mathbf{G}$. If the code is specified by its parity check matrix $\mathbf{H}$, the problem is to find if $\mathbf{H}$ has a sparse null vector.

Unfortunately, [24] proved that the problem of computing the minimum distance of an error-correcting code is NP-hard.

The problem of finding the minimum distance of a linear error-correcting code translates into finding a sparse null vector of the parity check matrix. Using the notation of Equation 2.2, finding sparse null vectors of structured matrices would solve the problem of finding sparse multiples of square-free matrices. Unfortunately, the hardness reduction of [24] does not seem to carry over for proving hardness for instances arising out of our problem.

4.3 Sparse vectors in integer lattices

Techniques similar to those used in [24] prove that the problem of finding sparse vectors in integer lattices is NP-complete, a problem that was stated to be open in [6]. We prove this result in Theorem 4.3. Again, this result comes very close to proving the hardness for finding sparse multiples over rationals, except for a last column.

***Sparse Lattice Vector*:** Given a lattice $\mathcal{L} \subseteq \mathbb{Z}^n$ specified by its basis vectors $\{b_1, \dots, b_n\}$ and an integer w, is there a lattice vector with Hamming weight at most w?

Theorem 4.3. *The problem* Sparse Lattice Vector *of finding a low Hamming weight lattice vector in an integer lattice specified by its basis is* NP*-complete.*

Proof. To see that the problem is in NP, a non-deterministic machine can guess the positions at which the lattice vector is nonzero. All that remains is to remove the corresponding rows and verify that the columns of the surviving submatrix are linearly dependent. The latter can be done deterministically using standard linear algebra algorithms. In the following, we give a NP-hardness proof.

We give a Cook-reduction from the problem *Subset Sum*, a well-known NP-complete problem.

The standard formulation of *Subset Sum* over integers is as follows:

***Subset Sum*:** Given distinct integers $\{z_1, \ldots, z_n\}$, a target integer t and a positive integer $w \leq n$, is there a non-empty subset $S \subseteq \{1, \ldots, n\}$ of size exactly w such that such that $\sum_{i \in S} z_i = t$?

Given an instance $\{z_1, \ldots, z_n\}$ of subset sum, to check if there is a subset of size w summing to t, the reduction first creates the following matrix:

$$M_w = \begin{bmatrix} 1 & 1 & \cdots & 1 & 0 \\ z_1 & z_2 & \cdots & z_n & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ z_1^{w-1} & z_2^{w-1} & \cdots & z_n^{w-1} & 1 \\ z_1^{w} & z_2^{w} & \cdots & z_n^{w} & t \end{bmatrix} \in \mathbb{Z}^{(w+1)\times(n+1)}. \tag{4.1}$$

Lemma 4.4 (stated and proved below) shows that M_w has a null vector of sparsity at most $w+1$ if and only if $z_{i_1} + z_{i_2} + \cdots + z_{i_w} = t$ for some $i_1 < i_2 < \ldots < i_w$.

To create an instance of *Sparse Lattice Vector*, the reduction creates a matrix N such that the columns of N span the nullspace of M (see Lemma 2.1). The instance $(\mathcal{L}, w)$, where $\mathcal{L}$ is the column lattice $\mathcal{L}$ of N, is fed to an algorithm claiming to solve the *Sparse Vector Problem.* □

Lemma 4.4. *The matrix M_w from* (4.1) *has a null vector of Hamming weight $w+1$ if and only if $z_{i_1} + z_{i_2} + \cdots + z_{i_w} = t$ for some $i_1 < i_2 < \ldots < i_w$.*

Proof. The row-rank of M_w is $w+1$ since they contain a Vandermonde submatrix with distinct z_i. Therefore any null vector has sparsity at least $w+1$. Consider a $(w+1)$-sized subset of columns. If the last column is not in this set, the chosen columns form a Vandermonde matrix. Since z_i are all distinct, the determinant of this Vandermonde matrix is nonzero and hence there doesn't exist a null vector of sparsity $w+1$ excluding the last column. Therefore assume that the last column is among those chosen, the determinant of the resulting matrix can be expanded as:

$$\begin{vmatrix} 1 & \cdots & 1 & 0 \\ z_{i_1} & \cdots & z_{i_w} & 0 \\ \vdots & & \vdots & \vdots \\ z_{i_1}^{w-1} & \cdots & z_{i_w}^{w-1} & 1 \\ z_{i_1}^{w} & \cdots & z_{i_w}^{w} & t \end{vmatrix} = t \begin{vmatrix} 1 & 1 & \cdots & 1 \\ z_{i_1} & z_{i_2} & \cdots & z_{i_w} \\ \vdots & \vdots & \vdots & \vdots \\ z_{i_1}^{w-1} & z_{i_2}^{w-1} & \cdots & z_{i_w}^{w-1} \end{vmatrix} - \begin{vmatrix} 1 & \cdots & 1 \\ z_{i_1} & \cdots & z_{i_w} \\ \vdots & \vdots & \vdots \\ z_{i_1}^{w-2} & \cdots & z_{i_w}^{w-2} \\ z_{i_1}^{w} & \cdots & z_{i_w}^{w} \end{vmatrix}.$$

The first of the matrices on the right-hand side is a Vandermonde matrix, whose determinant is well-known to be $\prod_{i_j \leq i_k}(z_{i_j} - z_{i_k})$. The second matrix is a first-order alternant ([20]) whose determinant is known to be $(z_{i_1} + z_{i_2} + \cdots + z_{i_w}) \prod_{i_j \leq i_k}(z_{i_j} - z_{i_k})$. Hence the determinant of the entire matrix is $(t - z_{i_1} - z_{i_2} - \cdots - z_{i_w}) \prod_{i_j \leq i_k}(z_{i_j} - z_{i_k})$. Since all the z_i are distinct, the determinant vanishes if and only if the first term vanishes which holds when there exists a subset of $\{z_1, z_2, \ldots, z_n\}$ of size w summing to t. $\square$

Chapter 5

Binomial multiples over $\mathbb{Q}$

In this chapter we completely solve the problem of determining if there exists a binomial multiple of a rational input polynomial (i.e., a multiple of sparsity $t = 2$). That is, given input $f \in \mathbb{Q}[x]$ of degree d, we determine if there exists a binomial multiple $h = x^m - a \in \mathbb{Q}[x]$ of f, and if so, find such an h with minimal degree. The constant coefficient a will be given as a pair $(r, e) \in \mathbb{Q} \times \mathbb{N}$ representing $r^e \in \mathbb{Q}$. The algorithm requires a number of bit operations which is polynomial in d and $\log \mathcal{H}(f)$. Unlike the case of t-sparse multiples for $t \geq 3$ considered in a later chapter, no *a priori* bounds on the degree or height of h are required. We show that m may be exponential in d, and $\log a$ may be exponential in $\log \mathcal{H}(f)$, and give a family of polynomials with these properties.

Algorithm 5.1 begins by factoring the given polynomial $f \in \mathbb{Q}[x]$ into irreducible factors (using, e.g., the algorithm of [16]). We then show how to find a binomial multiple of each irreducible factor, and finally provide a combining strategy for the different multiples.

The following theorem of [21] characterizes binomial multiples of irreducible polynomials. Let $\phi(n)$ be Euler's totient function, the number of positive integers less than or equal to n which are coprime to n.

Fact 5.1 ([21], Proposition 4, Corollary 2.2)**.** *Let $f \in \mathbb{Q}[x]$ be irreducible of degree d. Suppose the least-degree binomial multiple of f (if one exists) is of degree m. Then there exist $n, t \in \mathbb{N}$ with $n \mid d$ and $\phi(t) \mid d$ such that $m = n \cdot t$.*

The following, easily derived from explicit bounds in [22], gives a polynomial bound on m.

Algorithm 5.1: Lowest degree Binomial Multiple of a Rational Polynomial

```
Input: $f \in \mathbb{Q}[x]$
Output: The lowest degree binomial multiple $h \in \mathbb{Q}[x]$ of $f$, or
        "NONE"
1  Factor $f$ into irreducible factors: $f = x^b f_1 \cdot f_2 \cdot f_u$
2  if $f$ is not squarefree then return "NONE"
3  for $i = 1, 2, 3, \ldots, u$ do
4      $d_i \leftarrow \deg f_i$
5      $m_i \leftarrow$ least $k \in \{d_i, \ldots, d_i \cdot (\lceil 3d_i \ln\ln d_i \rceil + 7)\}$ s.t. $x^k \operatorname{rem} f_i \in \mathbb{Q}$
6      if no such $m_i$ is found then return "NONE"
7      else $r_i \leftarrow x^{m_i} \operatorname{rem} f_i$
8  $m \leftarrow \operatorname{lcm}(m_1, \ldots, m_u)$
9  foreach 2-subset $\{i, j\} \subseteq \{1, \ldots, u\}$ do
10     if $|r_i|^{m_j} \neq |r_j|^{m_i}$ then return "NONE"
11     else if $\operatorname{sign}(r_i^{m/m_i}) \neq \operatorname{sign}(r_j^{m/m_j})$ then $m \leftarrow 2 \cdot \operatorname{lcm}(m_1, \ldots, m_u)$
12 return $x^b(x^m - r_1^{m/m_1})$, with $r_1$ and $m/m_1$ given separately
```

Lemma 5.2. *For all integers $n \geq 2$, $\phi(\lceil 3n \ln\ln n \rceil + 7) > n$.*

Proof. Following [22], Fact 6.16 of [10] shows that for all $n \geq 3$

$$\phi(n) > \frac{0.56146 \cdot n}{\ln\ln n} \cdot \left(1 - \frac{1.41}{\ln^2 \ln n}\right).$$

It is then easily derived (say using Maple) that $\phi(n) > 0.4n/\ln\ln(n)$ for $n \geq 9428$. We similarly note that

$$\frac{0.4(3n \ln\ln n)}{\ln\ln(3n \ln\ln n)} > n$$

for $n \geq 77$, and that the left hand side is an increasing function of n in this range. Thus

$$\phi(\lceil 3n \ln\ln(n) \rceil) \geq \frac{0.4(3n \ln\ln n)}{\ln\ln(3n \ln\ln n)} > n$$

for $n \geq 9428$. The inequality is verified mechanically (say using Maple) for $2 \leq n \leq 9428$. □

Combining Fact 5.1 with Lemma 5.2, we obtain the following explicit upper bound on the maximum degree of a binomial multiple of an irreducible polynomial.

Theorem 5.3. *Let $f \in \mathbb{Q}[x]$ be irreducible of degree d. If a binomial multiple of f exists, and has minimal degree m, then $m \le d \cdot (\lceil 3d \ln\ln d \rceil + 7)$.*

Proof. By Fact 5.1, $m = n \cdot t$ such that $n \mid d$ and $\phi(t) \mid d$. Define $\xi(n) = \lceil 3n \ln\ln n \rceil + 7$, and define $\xi^{-1}(n)$ to be the smallest integer such that $\xi(\xi^{-1}(n)) \ge n$. From Lemma 5.2, we have that $\phi(\xi(n)) > n$ for $n \ge 2$. Hence, $d \ge \phi(t) \ge \xi^{-1}(t)$. Since ξ is a non-decreasing function, $d \ge \xi^{-1}(t)$ implies that $\xi(d) \ge t$. Thus $m = n \cdot t \le d \cdot \xi(d) \le d \cdot (\lceil 3d \ln\ln d \rceil + 7)$. □

The above theorem ensures that for an irreducible f_i, Step 5 of Algorithm 5.1 computes the least-degree binomial multiple $x^{m_i} - r_i$ if it exists, and otherwise correctly reports failure. It clearly runs in polynomial time.

If f has any repeated factor, then it cannot have a binomial multiple (see Lemma 6.1 below). So assume the factorization of f is as computed in Step 1, and moreover f is squarefree. If any factor does not have a binomial multiple, neither can the product. If every irreducible factor does have a binomial multiple, Step 5 computes the one with the least degree. The following relates the degree of the minimal binomial multiple of the input polynomial to those of its irreducible factors.

Lemma 5.4. *Let $f \in \mathbb{Q}[x]$ be such that $f = f_1 \cdots f_u \in \mathbb{Q}[x]$ for distinct, irreducible $f_1, \ldots, f_u \in \mathbb{Q}[x]$. Let $f_i \mid x^{m_i} - r_i$ for minimal $m_i \in \mathbb{N}$ and $r_i \in \mathbb{Q}$, and let $f \mid x^m - r$ for $r \in \mathbb{Q}$. Then $\operatorname{lcm}(m_1, \ldots, m_u) \mid m$.*

Proof. It suffices to prove that if $f \mid x^m - r$ and $f_i \mid x^{m_i} - r_i$ for minimal m_i then $m_i \mid m$ since any multiple of f is also a multiple of f_i.

Assume for the sake of contradiction that $m = cm_i + \ell$ for $0 < \ell < m_i$. Then for any root $\alpha \in \mathbb{C}$ of f_i, we have that $r = \alpha^m = \alpha^{cm_i} \cdot \alpha^\ell = r_i^c \cdot \alpha^\ell$. Since r and r_i are both rational, so is α^ℓ. Also $\alpha^\ell = \beta^\ell$ for any two roots $\alpha, \beta \in \mathbb{C}$ of f_i. Hence $f_i \mid x^\ell - \alpha^\ell$ and $\ell < m_i$, contradicting the minimality of m_i.

Thus $m_i \mid m$, and therefore $\operatorname{lcm}(m_1, \ldots, m_u) \mid m$. □

Lemma 5.5. *For a polynomial $f \in \mathbb{Q}[x]$ factored into distinct irreducible factors $f = f_1 f_2 \ldots f_u$, with $f_i \mid x^{m_i} - r_i$ for $r_i \in \mathbb{Q}$ and minimal such m_i, a binomial multiple of f exists if and only if $|r_i|^{m_j} = |r_j|^{m_i}$ for every pair $1 \le i, j \le u$. If a binomial multiple exists, the least-degree binomial multiple of f is $x^m - r_i^{m/m_i}$ such that m either equals the least common multiple of the m_i or twice that number. It can be efficiently checked which of these cases holds.*

Proof. Let $\alpha_i \in \mathbb{C}$ be a root of f_i. For any candidate binomial multiple $x^m - r$ of f, we have (from Lemma 5.4) that $m_i \mid m$.

First, suppose that such a binomial multiple exists: $f \mid x^m - r$ with $r \in \mathbb{Q}$. It is easily seen from $\alpha_i^m = r$ and $\alpha_i^{m_i} = r_i$ that $r_i^{m/m_i} = r$. Since this holds for any f_i, we see that $r_i^{m/m_i} = r = r_j^{m/m_j}$ for any $1 \le i, j \le u$. Thus $|r_i|^{m_j} = |r_j|^{m_i}$ must hold.

Conversely, suppose that $|r_i|^{m_j} = |r_j|^{m_i}$ holds for every pair $1 \le i, j \le u$. We get that $|\alpha_i|^{l m_i m_j} = |\alpha_j|^{l m_j m_i}$, and hence $\left|\alpha_i^l\right| = \left|\alpha_j^l\right|$ for $l = \operatorname{lcm}(m_1, \ldots, m_u)$. But α_i^l are all rational since $m_i \mid l$. Thus $\alpha_i^{2l} = \alpha_j^{2l}$ for every pair i, j . Thus, there exists a binomial multiple of the original polynomial of degree $2l$.

To check whether $\alpha_i^l = \alpha_j^l$ holds (or in other words if the degree of the binomial multiple is actually the lcm), it suffices to check whether the sign of each α_i^l is the same. This is equivalent to checking whether the sign of each r_i^{l/m_i} is the same. Since we can explicitly compute l and all the r_i, the sign of each r_i^{l/m_i} can be easily computed from the sign of r_i and the parity of l/m_i.

$\square$

It is easy to see that Algorithm 5.1 performs polynomially many arithmetic operations. The following is, then, an easy consequence of Lemma 5.5.

Theorem 5.6. *Given a polynomial $f \in \mathbb{Q}[x]$, Algorithm 5.1 outputs the least-degree binomial multiple $x^m - r_i^{m/m_i}$ (with r_i and m/m_i output separately) if one exists or correctly reports the lack of a binomial multiple otherwise. Furthermore, it runs in deterministic time $(d + \mathcal{H}(f))^{O(1)}$.*

The constant coefficient of the binomial multiple cannot be output in standard form, but must remain an unevaluated power. Polynomials exist whose minimal binomial multiples have exponentially sized degrees and heights.

Theorem 5.7. *For any $d \geq 841$ there exists a polynomial $f \in \mathbb{Z}[x]$ of degree at most $d \log d$ whose minimal binomial multiple $x^m - a$ is such that $m > \exp(\sqrt{d})$. Also, $\mathcal{H}(f) \leq \exp(2d \log d)$ and $\mathcal{H}(a) > 2^{\exp(\sqrt{d})}$.*

Proof. We construct the family from a product of cyclotomic polynomials. Let $p_i \in \mathbb{N}$ be the i^{th} largest prime, and let $\Phi_{p_i} = (x^{p_i} - 1)/(x-1) \in \mathbb{Z}[x]$ be the ${p_i}^{\text{th}}$ cyclotomic polynomials (whose roots are the primitive ${p_i}^{\text{th}}$ roots of unity). This is well known to be irreducible in $\mathbb{Q}[x]$.

Let $\ell = \sqrt{2d}$ and $g = \prod_{1 \leq i \leq \ell} \Phi_{p_i}$. Then, using the fact easily derived from [22] that $i \log i < p_i < 1.25 i \log i$ for all $i \geq 25$ and verifying that $(p_i - 1) \leq 1.5 i \log i$ mechanically for smaller values of i,

$$\deg g = \sum_{1 \leq i \leq \ell} (p_i - 1) \geq \sum_{1 \leq i \leq \ell} i = \frac{l(l+1)}{2} \geq d,$$

and

$$\deg g = \sum_{1 \leq i \leq \ell} (p_i - 1) \leq \sum_{1 \leq i \leq \ell} 1.5 i \log i \leq 1.5 \left(\frac{\ell^2 + \ell}{2} \log \ell \right) \leq d \log d,$$

The degree m of the minimal binomial multiple is the lcm of the order of the roots, and hence equal to the product of primes less than or equal to p_ℓ. This is $\exp(\vartheta(p_\ell))$ (where ϑ is the Chebyshev theta function), and for $\ell \geq 41$

$$m \geq \exp(\vartheta(p_\ell)) \geq \exp(\vartheta(\ell)) \geq \exp\left(\ell\left(1 - \frac{1}{\log \ell}\right)\right) \geq \exp\left(\sqrt{d}\right),$$

for $d \geq 841$, again by [22].

Now let $f = g(2x)$, so the minimal binomial multiple of f is $x^m - 1/2^m$. We have that

$$\mathcal{H}(g) \leq \prod_{1 \leq i \leq \ell} (1 + p_i) \leq 2^\ell \prod_{1 \leq i \leq \ell} p_i \leq \exp(2\ell \log \ell)$$

and

$$\mathcal{H}(f) \leq 2^{\deg(g)} \mathcal{H}(g) \leq 2^{d \log d} \exp(d \log d + 2\sqrt{2d} \log \sqrt{2d}) \leq \exp(2d \log d).$$

for all ≥ 841. □

Chapter 6

t-sparse multiples over $\mathbb{Q}$

We examine the problem of computing t-sparse multiples of rational polynomials, for any fixed positive integer t. As with other types of polynomial computations, it seems that cyclotomic polynomials behave quite differently from cyclotomic-free ones. Accordingly, we first examine the case that our input polynomial f consists only of cyclotomic or cyclotomic-free factors. Then we see how to combine them, in the case that none of the cyclotomic factors are repeated.

Specifically, we will show that, given any rational polynomial f which does not have repeated cyclotomic factors, and a height bound $c \in \mathbb{N}$, we can compute a sparsest multiple of f with height at most c, or conclude that none exists, in time polynomial in the size of f and $\log c$ (but exponential in t).

First, notice that multiplying a polynomial by a power of x does not affect the sparsity, and so without loss of generality we may assume all polynomials are relatively prime to x; we call such polynomials *non-original* since they do not pass through the origin.

6.1 The cyclotomic case

Suppose the input polynomial f is a product of cyclotomic factors, and write the complete factorization of f as

$$f = \Phi_{i_1}^{e_i} \cdot \Phi_{i_2}^{e_2} \cdots \Phi_{i_k}^{e_k}, \tag{6.1}$$

where Φ_j indicates the j^{th} cyclotomic polynomial, the i_j's are all distinct, and the e_i's are positive integers.

Now let $m = \operatorname{lcm}(i_1, \ldots, i_k)$. Then m is the least integer such that $\Phi_{i_1} \cdots \Phi_{i_k}$ divides $x^m - 1$. Let $\ell = \max_i e_i$, the maximum multiplicity of any factor of f. This means that $(x^m - 1)^\ell$ is an $(\ell + 1)$-sparse multiple of f. To prove that this is in fact a sparsest multiple of f, we first require the following simple lemma. Here and for the remainder, for a univariate polynomial $f \in \mathsf{F}[x]$, we denote by f' the first derivative with respect to x, that is, $\frac{\mathrm{d}}{\mathrm{d}x} f$.

Lemma 6.1. *Let $h \in \mathbb{Q}[x]$ be a t-sparse and non-original polynomial, and write $h = a_1 + a_2 x^{d_2} + \cdots + a_t x^{d_t}$. Assume the complete factorization of h over $\mathbb{Q}[x]$ is $h = a_t h_1^{e_1} \cdots h_k^{e_k}$, with each h_i monic and irreducible. Then* $\max_i e_i \leq t - 1$.

Proof. Without loss of generality, assume h is *exactly* t-sparse, and each $a_i \neq 0$.

The proof is by induction on t. If $t = 1$ then $h = a_1$ is a constant, so $\max_i e_i = 0$ and the statement holds. Otherwise, assume the statement holds for $(t - 1)$-sparse polynomials.

Write the so-called "sparse derivative" $\tilde{h}$ of h as

$$\tilde{h} = \frac{h'}{x^{d_2 - 1}} = a_2 d_2 + a_3 d_3 x^{d_3 - d_2} + \cdots + a_{t-1} d_{t-1} x^{d_{t-1} - d_2}.$$

For any i with $e_i > 0$, we know that $h_i^{e_i - 1}$ divides $\frac{d}{dx} h$, and h_i is relatively prime to $x^{d_2 - 1}$ since the constant coefficient of h is nonzero. Therefore $h_i^{e_i - 1}$ divides $\tilde{h}$. By the inductive hypothesis, since $\tilde{h}$ is $(t - 1)$-sparse and non-original, $e_i - 1 \leq t - 2$, and therefore $e_i \leq t - 1$. Since i was chosen arbitrarily, $\max_i e_i \leq t - 1$.

□

An immediate consequence is the following:

Theorem 6.2. *Let $f \in \mathbb{Q}[x]$ be a product of cyclotomic polynomials, written as in* (6.1). *Then*

$$h = (x^{\operatorname{lcm}(i_1, \ldots, i_k)} - 1)^{\max_i e_i}$$

is a sparsest multiple of f.

Proof. Clearly h is a multiple of f with exactly $\max_i e_i + 1$ nonzero terms. By way of contradiction, suppose a $(\max_i e_i)$-sparse multiple of f exists; call it $\bar{h}$. Without loss of generality, we can assume that $\bar{h}$ is non-original. Then from Lemma 6.1, the maximum multiplicity of any factor of $\bar{h}$ is $\max_i e_i - 1$. But this contradicts the fact that each $\Phi_i^{e_i}$ must divide $\bar{h}$. Therefore the original statement is false, and every multiple of f has at least $\max_i e_i + 1$ nonzero terms. □

6.2 The cyclotomic-free case

We say a polynomial $f \in \mathbb{Q}[x]$ is *cyclotomic-free* if it contains no cyclotomic factors. Here we will show that a sparsest multiple of a cyclotomic-free polynomial must have degree bounded by a polynomial in the size of the input and output.

First we need the following elementary lemma.

Lemma 6.3. *Suppose $f, h \in \mathbb{Q}[x]$ with f irreducible, and k is a positive integer. Then $f^k | h$ if and only if $f | h$ and $f^{k-1} | h'$.*

Proof. The $\Rightarrow$ direction is straightforward.

For the $\Leftarrow$ direction, suppose $f|h$ and $f^{k-1}|h'$. Let ℓ be the maximum multiplicity of f in h, and write $h = f^\ell g$ with $g \in \mathbb{Q}[x]$ relatively prime to f.

We can write $h' = f^{\ell-1}(fg' + \ell f' g)$. Now, by way of contradiction, assume that $k > \ell$. Then f divides $fg' + \ell f' g$, and therefore f divides $\ell f' g$. But this is impossible from the assumption that f is irreducible and relatively prime to g. Therefore $k \le \ell$, and $f^k | f^\ell | h$. □

The following technical lemma provides the basis for our degree bound on the sparsest multiple of a non-cyclotomic polynomial.

Lemma 6.4. *Let $f, h_1, h_2, \ldots, h_\ell \in \mathbb{Q}[x]$ be non-original polynomials, where f is irreducible and non-cyclotomic with degree d, and each h_i satisfies* $\deg h_i \le$ *u and $\mathcal{H}(h_i) \le c$. Also let $k, m_1, m_2, \ldots, m_\ell$ be positive integers such that*

$$f^k | (h_1 x^{m_1} + h_2 x^{m_2} + \cdots + h_\ell x^{m_\ell}).$$

Then f^k divides each h_i whenever every "gap length", for $1 \leq i < \ell$, satisfies

$$m_{i+1} - m_i - \deg h_i \geq \frac{1}{2} d \cdot \ln^3(3d) \cdot \ln\left(u^{k-1} c\,(t-1)\right). \tag{6.2}$$

Proof. The proof is by induction on k. For the base case, let $k = 1$. Then we have a separate, inner induction on ℓ. The inner base case, when $k = \ell = 1$, is clear since f is non-original. Now assume the lemma holds whenever $k = 1$ and $1 \leq \ell - 1 < r$ for some $r \geq 2$. Let $g_1 = h_1 x^{m_1}$ and $g_2 = h_2 + \cdots + h_\ell x^{m_r - m_2}$, so that $f \mid (g_1 + g_2 x^{m_2})$. Since

$$m_2 - \deg g_1 \geq \frac{1}{2} d \cdot \ln^3(3d) \cdot \ln(c(t-1)),$$

we can apply [17, Proposition 2.3] to conclude that $f \mid g_1$ and $f \mid g_2$. This means $f \mid h_1$ and, by the inner induction hypothesis, $f \mid h_i$ for $2 \leq i \leq \ell$ as well. Therefore the lemma holds whenever $k = 1$.

Now assume the lemma holds whenever $\ell \geq 1$ and $1 \leq k < s$, for some $s \geq 2$. Next let ℓ be arbitrary and $k = s$. So we write $f^s | (h_1 x^{m_1} + \cdots + h_\ell x^{m_\ell})$.

The derivative of the right hand side is

$$h_1' x^{m_1} + m_1 h_1 x^{m_1 - 1} + \cdots + h_\ell' x^{m_\ell} + m_\ell h_\ell x^{m_\ell - 1},$$

which must be divisible by f^{s-1}. But by the induction hypothesis, f^{s-1} also divides each h_i, so we can remove all terms with h_i from the previous formula and conclude that $f^{s-1} | \, (h_1' x^{m_1} + \cdots + h_\ell' x^{m_\ell})$.

Since each $\mathcal{H}(h_i) \leq c$ and $\deg h_i \leq u$, the height of the derivative satisfies $\mathcal{H}(h_i') \leq uc$. A second application of the induction hypothesis therefore shows that each h_i' is divisible by f^{s-1}. Since $s - 1 \geq 1$, we already know that each h_i is divisible by f, and then applying Lemma 6.3 completes the proof. □

Our main tool in proving that Algorithm 2.1 is useful for computing the sparsest multiple of a rational polynomial, given only a bound c on the height, in polynomial time in the size of f and $\log c$, is the following degree bound on the sparsest height-bounded multiple of a rational polynomial.

Theorem 6.5. *Let $f \in \mathbb{Q}[x]$ with $\deg f = d$ be cyclotomic-free, and let $t, c \in \mathbb{N}$ such that f has a nonzero t-sparse multiple with height at most c. Denote by n the smallest degree of any such multiple of f. Then n satisfies*

$$n \leq 2(t-1)B \ln B, \tag{6.3}$$

where B is the formula polynomially bounded by d, $\log c$, and $\log t$ defined as

$$B = \frac{1}{2} d^2 \cdot \ln^3(3d) \cdot \ln\left(\hat{c}\,(t-1)^d\right), \tag{6.4}$$

and $\hat{c} = \max(c, 35)$.

Proof. Let h be a t-sparse multiple of f with degree n and height $\mathcal{H}(h) \leq c$. Without loss of generality, assume $d \geq 1$, $t \geq 2$, and both f and h are non-original.

By way of contradiction, assume $n > 2(t-1)B \ln B$. For any univariate polynomial define the *gap lengths* to be the differences of consecutive exponents of nonzero terms. Split h at every gap greater than $2B \ln B$ by writing

$$h = h_1 x^{m_1} + h_2 x^{m_2} + \cdots + h_\ell x^{m_\ell},$$

where each $h_i \in \mathbb{Q}[x]$ has nonzero constant term and each gap length satisfies $m_{i+1} - m_i - \deg h_i > 2B \ln B$. Since we split h at *every* sufficiently large gap, and h has at most t nonzero terms, each h_i has degree at most $u = 2(t-1)B \ln B$.

We want to show that the gap length $2B \ln B$ is sufficiently large to apply Lemma 6.4. For this, first notice that $2B \ln B = B \ln(B^2)$. Since B is positive, $B^2 > 2B \ln B$, so the gap length is greater than $B \ln(2B \ln B)$.

Since $\hat{c} \geq 35$, $B \geq 2.357$, and then

$$(d-1)\ln(2B\ln B)\cdot\ln(\hat{c}(t-1)^d) > \ln\left((2B\ln B)^{d-1} \cdot \hat{c}(t-1)^d\right) = \ln\left(u^{d-1}\hat{c}\,(t-1)\right).$$

Then from the definition of B in (6.4), the gap length satisfies

$$2B \ln B > B \ln(2B \ln B) > \frac{1}{2} d \cdot \ln^3(3d) \cdot \ln\left(u^{d-1}\hat{c}\,(t-1)\right).$$

Finally, notice that the maximum multiplicity of any factor of f is at most $\deg f = d$. (So, using the notation of Lemma 6.4, $d \geq k$.) Therefore

Lemma 6.4 applies to each factor of f (to full multiplicity) and we conclude that f divides each h_i.

But then, since there is at least one gap and $\ell > 1$, h_1 is a multiple of f with fewer terms and lower degree than h. This is a contradiction, so we are done. □

In order to compute the sparsest multiple of a rational polynomial with no cyclotomic or repeated factors, we therefore can simply call Algorithm 2.1 with the given height bound c and degree bound as specified in (6.3).

6.3 Handling cyclotomic factors

Suppose f is any non-original rational polynomial with no repeated cyclotomic factors. Factor f as $f = f_C \cdot f_D$, where f_C is a squarefree product of cyclotomics and f_D is cyclotomic-free. Write the factorization of f_C as $f_C = \Phi_{i_1} \cdots \Phi_{i_k}$, where Φ_n is the n^{th} cyclotomic polynomial. Since every i^{th} root of unity is also a $(mi)^{\text{th}}$ root of unity for any $m \in \mathbb{N}$, f_C must divide the binomial $x^{\text{lcm}\{i_1,\ldots,i_k\}} - 1$, which is in fact a sparsest multiple of f_C (Theorem 6.2) and clearly has minimal height.

Then we will show that a sparsest height-bounded multiple of f is either of small degree, or can be constructed as a sparsest height-bounded multiple of f_D times the binomial multiple of f_C specified above. Algorithm 6.1 uses this fact to compute a sparsest multiple of any such f.

Theorem 6.6. *Let $f \in \mathbb{Q}[x]$ be a degree-d non-original polynomial with no repeated cyclotomic factors. Given f and integers c and t, Algorithm 6.1 correctly computes a t-sparse multiple h of f satisfying $\mathcal{H}(h) \leq c$, if one exists. The sparsity of h will be minimal over all multiples with height at most c. The cost of the algorithm is $(d \cdot \log \mathcal{H}(f) \cdot \log c)^{O(t)}$.*

Proof. Step 1 can be accomplished in the stated complexity bound using [16]. The cost of the remaining steps follows from basic arithmetic and Theorem 2.3. Define h to be sparsest multiple of f of least degree that satisfies $\mathcal{H}(h) \leq c$. We have two cases:

Algorithm 6.1: Rational Sparsest Multiple

```
    Input: Bounds $t, c \in \mathbb{N}$ and $f \in \mathbb{Q}[x]$ a non-original polynomial of
           degree $d$ with no repeated cyclotomic factors
    Output: $t$-sparse multiple $h$ of $f$ with $\mathcal{H}(h) \leq c$, or "NONE"
1   Factor $f$ as $f = \Phi_{i_1} \cdot \Phi_{i_2} \cdots \Phi_{i_k} \cdot f_D$, where $f_D$ is cyclotomic-free
2   $n \leftarrow$ degree bound from (6.3)
3   $\hat{h} \leftarrow \lfloor t/2 \rfloor$-sparse multiple of $f_D$ with $\mathcal{H}(\hat{h}) \leq c$ and $\deg \hat{h} \leq n$, using
    Algorithm 2.1
4   $\tilde{h} \leftarrow t$-sparse multiple of $f$ with $\mathcal{H}(h) \leq c$ and $\deg h \leq n$, using
    Algorithm 2.1
5   if $\hat{h} =$"NONE" and $\tilde{h} =$"NONE" then return "NONE"
6   else if $\hat{h} =$"NONE" or sparsity($\tilde{h}$) $\leq 2 \cdot$ sparsity($\hat{h}$) then return $\tilde{h}$
7   $m \leftarrow \operatorname{lcm}\{i_1, i_2, \ldots, i_k\}$
8   return $\hat{h} \cdot (x^m - 1)$
```

Case 1: $\deg h \leq n$. Then the computed $\tilde{h}$ must equal h. Furthermore, since this is the sparsest multiple, either $\hat{h}$ does not exist or the sparsity of $\hat{h}$ is greater than or equal to the sparsity of $\tilde{h}$. So $h = \tilde{h}$ is correctly returned by the algorithm in this case.

Case 2: $\deg h > n$. Then, using Lemma 6.4, since $f_D \mid h$, h can be written $h = h_1 + x^i h_2$, for some $i > \deg h_1$, and f_D divides both h_1 and h_2. By Theorem 2.3, sparsity($\hat{h}$) must then be less than or equal to each of sparsity(h_1) and sparsity(h_2). But since sparsity(h) = sparsity(h_1) + sparsity(h_2), this means that the sparsity of $\hat{h} \cdot (x^m - 1)$ is less than or equal to the sparsity of h, and hence this is a sparsest multiple.

□

6.4 An example

Say we want to find a sparsest multiple, with coefficients at most 1000 in absolute value, of the following polynomial over $\mathbb{Z}[x]$:

$$f = x^{10} - 5x^9 + 10x^8 - 8x^7 + 7x^6 - 4x^5 + 4x^4 + x^3 + x^2 - 2x + 4.$$

Note that finding the *sparsest* multiple would correspond to setting $t = 10$ in the algorithm (since the least-degree 11-sparse multiple is f itself). To accomplish this, we first factor f using [16] and identify cyclotomic factors:

$$f = \underbrace{(x^2 - x + 1)}_{\Phi_6} \cdot \underbrace{(x^4 - x^3 + x^2 - x + 1)}_{\Phi_{10}} \cdot \underbrace{(x^4 - 3x^3 + x^2 + 6x + 4)}_{f_D}.$$

Next, we calculate a degree bound from Theorem 6.5. Unfortunately, this bound is not very tight (despite being polynomial in the output size); using $t = 10$, $c = 1000$, and f given above, the bound is $n \leq 11\,195\,728$. So for this example, we will use the smaller (but artificial) bound of $n \leq 20$.

The next step is to calculate the sparsest 5-sparse multiple of f_D and 10-sparse multiple of f with degrees at most 20 and heights at most 1000. Using Algorithm 2.1, these are respectively

$$\begin{aligned}\hat{h} &= x^{12} + 259x^6 + 64,\\ \tilde{h} &= x^{11} - 3x^{10} + 12x^8 - 9x^7 + 10x^6 - 4x^5 + 9x^4 + 3x^3 + 8.\end{aligned}$$

Since the sparsity of $\hat{h}$ is less than half that of $\tilde{h}$, a sparsest multiple is

$$\begin{aligned}h &= (x^{12} + 259x^6 + 64) \cdot (x^{\operatorname{lcm}(6,10)} - 1)\\ &= x^{42} + 259x^{36} + 64x^{30} - x^{12} - 259x^6 - 64\end{aligned}$$

Chapter 7

Sparse multiples over $\mathbb{F}_q$

We prove that for any constant t, finding the minimal degree t-sparse multiple of an $f \in \mathbb{F}_q[x]$ is harder than finding orders of elements in $\mathbb{F}_{q^e}$. The latter problem is thought to be hard, essentially of the same complexity as factoring integers. In the second section, we tightly characterize (up to randomization) the complexity of finding binomial multiples over finite fields. In the last section, we discuss connections of the problem with those considered in cryptographic settings.

7.1 Hardness of t-sparse multiple finding

Formal problem definitions are as follows:

$\mathbf{SpMul}^{(t)}_{\mathbb{F}_q}(f, n)$: Given a polynomial $f \in \mathbb{F}_q[x]$ and an integer $n \in \mathbb{N}$, determine if there exists a (nonzero) 2-sparse multiple $h \in \mathbb{F}_q[x]$ of f with $\deg h \leq n$.

$\mathbf{Order}_{\mathbb{F}_{q^e}}(a, n)$: Given an element $a \in \mathbb{F}^*_{q^e}$ and an integer $n < q^e$, determine if there exists a positive integer $m \leq n$ such that $a^m = 1$.

The problem $\mathbf{Order}_{\mathbb{F}_{q^e}}(a, n)$ is well-studied (see for instance [19]), and has been used as a primitive in several cryptographic schemes. Note that an algorithm to solve $\mathbf{Order}_{\mathbb{F}_{q^e}}(a, n)$ will allow us to determine the *multiplicative* *order* of any $a \in \mathbb{F}_{q^e}^*$ (the smallest nonzero m such that $a^m = 1$) with essentially the same cost (up to a factor of $O(e \log q)$) by using binary search.

The reduction from $\mathbf{Order}_{\mathbb{F}_{q^e}}(a, n)$ to $\mathbf{SpMul}_{\mathbb{F}_q}^{(t)}(f, n)$ works as follows: Given an instance of $\mathbf{Order}_{\mathbb{F}_{q^e}}(a, n)$, we first check if the order o_a of a is less than t by brute-force. Otherwise, we construct the minimal polynomial g_{a^i} (over $\mathbb{F}_q$) for each $a^0, a^1, a^2, \ldots, a^{t-1}$. We only keep distinct g_{a_i}, and call the product of these distinct polynomials $f_{a,t}$. We then run the $\mathbf{SpMul}_{\mathbb{F}_q}^{(t)}(f, n)$ subroutine to search for the existence of a degree n, t-sparse multiple of the polynomial $f_{a,t}$.

Theorem 7.1. *Let $a \in \mathbb{F}_q$ be an element of order at least t. Then the least degree t-sparse multiple of $f_{a,t}$ is $x^{o_a} - 1$ where o_a is the order of a.*

Proof. It is easy to see that $x^{o_a} - 1$ is a multiple of the given polynomial. We need to prove that it is actually the least-degree t-sparse multiple.

By equation 2.2 in Chapter 2, a degree n multiple h of $f_{a,t}$ corresponds to the following set of linear equations:

$$\underbrace{\begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & a & a^2 & \cdots & a^{n-1} \\ 1 & a^2 & a^4 & \cdots & a^{2n-2} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & a^t & a^{2t} & \cdots & a^{tn-t} \end{bmatrix}}_{A(f_{a,t},n)} \begin{bmatrix} h_0 \\ h_1 \\ \\ \vdots \\ h_{n-1} \end{bmatrix} = 0.$$

To prove that no t-sparse multiple h of degree less than o_a exists, it suffices to show that any t columns of $A(f_{a,t}, o_a - 1)$ are linearly independent. Consider the $(t \times t)$-matrix corresponding to some choice of t columns:

$$B = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ a^{i_1} & a^{i_2} & \cdots & a^{i_t} \\ \vdots & \vdots & \vdots & \vdots \\ a^{ti_1} & a^{ti_2} & \cdots & a^{ti_t} \end{bmatrix}$$

This Vandermonde matrix $\mathbf{B}$ has determinant $\prod_{1 \le j < k \le t}(a^{i_k} - a^{i_j})$ which is nonzero since $i_j < i_k < o_a$ and hence $a^{i_j} \neq a^{i_k}$. Thus the least-degree t-sparse multiple of the given polynomial is $x^{o_a} - 1$.

□

Algorithm 7.1: Least degree binomial multiple of f over $\mathbb{F}_q$

```
Input: $f \in \mathbb{F}_q[x]$
Output: The least degree binomial multiple $h$ of $f$
1  Factor $f = x^b f_1^{e_1} \cdot f_2^{e_2} \cdot f_\ell^{e_\ell}$ for irreducible $f_1, \ldots, f_\ell \in \mathbb{F}_q[x]$, and set
   $d_i \leftarrow \deg f_i$
2  for $i = 1, 2, \ldots, \ell$ do
3      $a_i \leftarrow x \in \mathbb{F}_q[x]/(f_i)$, a root of $f_i$ in the extension $\mathbb{F}_{q^{d_i}}$
4      Calculate $o_i$, the order of $a_i$ in $\mathbb{F}_q[x]/(f_i)$.
5  $n_1 \leftarrow \mathrm{lcm}(\{o_i/\gcd(o_i, q-1)\})$ for all $i$ such that $d_i > 1$
6  $n_2 \leftarrow \mathrm{lcm}(\{order(a_i/a_j)\})$ over all $1 \leq i, j \leq u$
7  $n \leftarrow \mathrm{lcm}(n_1, n_2)$
8  $\tilde{h} \leftarrow (x^n - a_1^n)$
9  $e \leftarrow \lceil \log_p \max e_i \rceil$, the smallest $e$ such that $p^e \geq e_i$ for all $i$
10 return $h = x^b(x^n - a_1^n)^{p^e}$
```

7.2 Probabilistic algorithm for finding binomial multiples

Next we give a probabilistic algorithm for finding the least degree binomial multiple for polynomials $f \in \mathbb{F}_q$. This algorithm makes repeated calls to an $\mathbf{Order}_{\mathbb{F}_{q^e}}(a, n)$ (defined in the previous section) subroutine. Combined with the hardness result of the previous section (with t=2), this characterizes the complexity of finding least-degree binomial multiples in terms of the complexity of $\mathbf{Order}_{\mathbb{F}_{q^e}}(a, n)$, up to randomization.

Algorithm 7.1 solves the binomial multiple problem in $\mathbb{F}_q$ by making calls to an $\mathbf{Order}_{\mathbb{F}_{q^e}}(a, n)$ procedure that computes the order of elements in extension fields of $\mathbb{F}_q$. Thus $\mathbf{SpMul}^{(2)}_{\mathbb{F}_q}(f)$ reduces to $\mathbf{Order}_{\mathbb{F}_{q^e}}(a, n)$ in probabilistic polynomial time. Construction of an irreducible polynomial (required for finite field arithmetic) as well as the factoring step in the algorithm make it probabilistic.

Theorem 7.2. *Given $f \in \mathbb{F}_q[x]$ of degree d, Algorithm 7.1 correctly computes a binomial multiple h of f with least degree. It uses at most d^2 calls to a routine for order finding in $\mathbb{F}_{q^e}$, for various $e \leq d$, and $d^{O(1)}$ other operations in $\mathbb{F}_q$. It is probabilistic of the Las Vegas type.*

Proof. As a first step, the algorithm factors the given polynomial into irreducible factors. Efficient probabilistic algorithms for factoring polynomials over finite fields are well-known ([9]).

First, suppose the input polynomial f is irreducible, i.e. $\ell = e_1 = 1$ in Step 1. Then it has the form $f = (x - a)(x - a^q) \cdots (x - a^{q^{d-1}})$ for some $a \in \mathbb{F}_{q^d}$, where $d = \deg f$. If $f = (x - a)$, the least-degree binomial multiple is f itself. Therefore, assume that $d > 1$. Let the least-degree binomial multiple (in $\mathbb{F}_q[x]$) be $x^n - \beta$

Since both a and a^q are roots of $(x^n - \beta)$, we have that $a^n = a^{nq}$ and $a^{n(q-1)} = 1$. Thus, the order o_a of a divides $n(q - 1)$. The minimal n for which $o_a \mid n(q - 1)$ is $n = \frac{o_a}{\gcd(o_a, q-1)}$. Since this n ensures that $a^n = a^{nq}$, it also simultaneously ensures that each a^{q^i} is also a root.

Notice that this n equals n_1 computed on Step 5, and n_2 computed on Step 6 will equal 1, so the algorithm is correct in this case.

Now suppose the input polynomial f is reducible. The factorization step factors f into irreducible factors $f = f_1^{e_1} f_2^{e_2} \cdots f_\ell^{e_\ell}$. Let $\check{f} = f_1 f_2 \cdots f_\ell$ denote the squarefree part of f.

Being irreducible, each f_i has the form $f_i(x) = (x - a_i)(x - a_i^q) \cdots (x - a_i^{q^{d_i - 1}})$ for some $a_i \in \mathbb{F}_{q^d}$, and $d_i = \deg f_i$. We notice two facts:

- If $\check{f}(x) \mid x^n - a$ for some $a \in \mathbb{F}_q$, we have that $a_i^n = a_j^n$ for all $1 \leq i, j \leq \ell$, and hence that $(\frac{a_i}{a_j})^n = 1$. Thus $order(\frac{a_i}{a_j}) \mid n$. The least integer satisfying these constraints is n_2 computed on Step 6.

- As before for the case when the input polynomial is irreducible and of degree more than one: $d_i > 1$ implies that $\frac{o_i}{\gcd(o_i, q-1)} \mid n$ for o_i the order of a_i. The least integer satisfying these constraints is n_1 computed on Step 5.

The minimal n is the least common multiple of all the divisors obtained from the above two types of constraints, which is exactly the value computed on Step 7. The minimal degree binomial multiple of $\check{f}$ is $x^n - a_1^n$.

It is easily seen that for the smallest e such that $p^e \geq e_i$, $(x^n - a^n)^{p^e}$ is a binomial multiple of f. The following claim proves that it is actually the minimal degree binomial multiple.

Claim. Let e be the smallest non-negative integer such that $p^e \geq \max e_i$. Then the minimal degree binomial multiple of f is $(x^n - a_i^n)^{p^e}$ for n obtained as above.

Proof of claim. Let the minimal degree binomial multiple of f be $x^{\hat{n}} - b$. Factor $\hat{n}$ as $\hat{n} = \check{n}p^c$ for maximal c, and write $(x^{\hat{n}} - b)$ as $(x^{\check{n}} - b^{1/p^c})^{p^c}$. The squarefree part of f, $\check{f}$ divides $(x^{\check{n}} - b^{1/p^c})$, and hence (by constraints on and minimality of n) $(x^n - a_1^n) \mid (x^{\check{n}} - b^{1/p^c})$. Thus $\check{n} \geq n$.

Since c is chosen maximally, p does not divide $\check{n}$, and hence $x^{\check{n}} - b^{1/p^c}$ is squarefree. Using this and the fact that f divides $(x^{\check{n}} - b^{1/p^c})^{p^c}$, it is seen that $p^c \geq e_i$ holds for all e_i, and hence $p^c \geq p^e$. This, along with $\check{n} \geq n$, completes the proof.

□

7.3 Relationship to problems in cryptography

Problems concerning sparse multiples have been considered in the cryptography community to some extent. The problem arises in this area of research due to its use in breaking Linear Feedback Shift Register (LFSR) based stream ciphers.

For the scope of this presentation, it suffices to say that a LFSR is a mechanism for generating pseudo-random bits, each bit being generated from a fixed linear combination of the last few bits.

Stream ciphers based on LFSRs combine outputs of finite number of LFSRs using a nonlinear function. The recurrence of a LFSR can be described using a polynomial over F_2. To resist attacks, this polynomial is chosen to be a dense primitive polynomial. Furthermore, because of the nature of possible attacks, it is desirable that this polynomial not have a low-degree sparse multiple.

Some effort has been devoted in literature to reason about the least degree sparse multiples of primitive polynomials. [14, 12, 18] study trinomial multiples of primitive polynomials. They prove several results regarding the number of low-degree trinomial multiples, the structure of trinomial multiples and about trinomials having large number of primitive factors. They also give algorithms for finding sparse multiples.

Although the techniques used in these papers might be useful in gaining some intuition, the results do not seem to directly apply to our problem. This is for two reasons. Firstly, the results mostly concentrate on proving upper bounds on the degree of the least degree trinomial multiples of *all* polynomials. Thus, they fail to give any meaningful bounds for the case when the given polynomial has a trinomial multiple of degree much lesser than the average. Also, unfortunately, the algorithms presented are heuristic, and hence fail to provide any rigorous complexity guarantees.

In addition to the above discussion about the choice of the polynomial of the LFSR, another place where sparse multiples appear in the area's literature is correlation attacks on LFSR based stream ciphers. See for instance [15, 4]. These attacks require several sparse multiples of degree less than a large bound. However, since the algorithms for finding the sparse multiples employ algorithms for known hard problems (see [5, 7]), not much attention is devoted to optimizing the degree of the sparse multiples. Again, as before, the techniques are largely heuristic, and lack rigorous complexity guarantees.

It is a hope that completely characterizing complexities of sparsity problems over finite fields would help design better heuristics for the problem instances used in practice.

Chapter 8

Conclusion

In this thesis, we have exhibited an efficient algorithm to compute the least-degree binomial multiple of any rational polynomial. We have also shown how to compute t-sparse multiples of rational polynomials that do not have repeated cyclotomic factors, for any fixed t, and given a bound on the height of the multiple.

We have also shown that, even for fixed t, finding a t-sparse multiple of a degree-d polynomial over $\mathbb{F}_q[x]$ is at least as hard as finding the orders of elements in $\mathbb{F}_{q^d}$. In the $t = 2$ case, there is also a probabilistic reduction in the other direction, so that computing binomial multiples of degree-d polynomials over $\mathbb{F}_q[x]$ probabilistically reduces to order finding in $\mathbb{F}_{q^d}$.

Several important questions remain unanswered. Although we have an unconditional algorithm to compute binomial multiples of rational polynomials, computing t-sparse multiples for fixed $t \geq 3$ requires an a priori height bound on the output as well as the requirement that the input contains no repeated cyclotomic factors. Removing these restrictions is desirable (though not necessarily possible).

Open Problem 8.1. *For $t \geq 3$, remove the necessity of height bounds as part of the input.*

Our algorithm for finding t-sparse height-bounded multiples for rational polynomials, and the algorithm for finding binomial multiples of polynomials over finite fields use randomization. Over finite fields, algorithms for factoring polynomials give the algorithm its probabilistic nature. Over the rationals, the randomization is due to the subroutine for finding short l_∞ vector in fixed-dimension lattices. A very interesting question is to investigate whether these algorithms can be made deterministic. In particular, over the rationals, it seems plausible that a much simpler deterministic algorithm for finding short l_∞ vector exists.

Open Problem 8.2. *Can the randomized algorithms in this thesis be made deterministic?*

Regarding lower bounds, we know that computing t-sparse multiples over finite fields is at least as hard as order finding, a result which is tight (up to randomization) for $t = 2$, but for larger t we believe the problem is even harder. Specifically, we suspect that computing t-sparse multiples is NP-complete over both $\mathbb{Q}$ and $\mathbb{F}_q$, when t is a parameter in the input.

Open Problem 8.3. *If t is part of the input, is the problem of finding t-sparse multiples NP-hard?*

References

[1] Miklós Ajtai, Ravi Kumar, and D. Sivakumar. A sieve algorithm for the shortest lattice vector problem. In *Symp. Theory of Computing (STOC'01)*, pages 601–610, 2001. 8, 10

[2] E. R. Berlekamp, R. J. McEliece, and H. C. van Tilborg. On the inherent intractability of certain coding problems. *IEEE Transactions on Information Theory*, 24(3), 1978. 1, 19

[3] R. P. Brent and P. Zimmermann. Algorithms for finding almost irreducible and almost primitive trinomials. In *Primes and Misdemeanours: Lectures in Honour of the Sixtieth Birthday of Hugh Cowie Williams, Fields Institute*, page 212, 2003. 1

[4] Philippe Chose, Antoine Joux, and Michel Mitton. Fast correlation attacks: An algorithmic point of view. In *EUROCRYPT '02: Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques*, pages 209–221, London, UK, 2002. Springer-Verlag. 42

[5] F. Didier and Y. Laigle-Chapuy. Finding low-weight polynomial multiples using discrete logarithms. In *Proc. IEEE International Symposium on Information Theory (ISIT 2007)*, pages 1036–1040, 2007. 1, 42

[6] Sebastian Egner and Torsten Minkwitz. Sparsification of rectangular matrices. *J. Symb. Comput.*, 26(2):135–149, 1998. 1, 21

[7] Laila El Aimani and Joachim von zur Gathen. Finding low weight polynomial multiples using lattices. Cryptology ePrint Archive, Report 2007/423, 2007. `http://eprint.iacr.org/2007/423.pdf`. 1, 42

[8] I. Z. Emiris and I. S. Kotsireas. Implicitization exploiting sparseness. In *Geometric and algorithmic aspects of computer-aided design and manufacturing*, volume 67 of *DIMACS Ser. Discrete Math. Theoret. Comput. Sci.*, pages 281–297, 2005. 1

[9] Joachim von zur Gathen and J. Gerhard. *Modern Computer Algebra*, chapter 14, pages 367–380. Cambridge University Press, New York, NY, USA, 2003. 40

[10] Mark Giesbrecht. *Nearly Optimal Algorithms for Canonical Matrix Forms*. PhD thesis, University of Toronto, 1993. 196 pp. 25

[11] Mark Giesbrecht, Daniel S. Roche, and Hrushikesh Tilak. Computing sparse multiples of polynomials. Submitted to *International Symposium on Algorithms and Computation*, 2010. iv, 3

[12] Kishan Chand Gupta and Subhamoy Maitra. Primitive polynomials over gf(2) - a cryptologic approach. In *ICICS '01: Proceedings of the Third International Conference on Information and Communications Security*, pages 23–34, London, UK, 2001. Springer-Verlag. 41

[13] Venkatesan Guruswami and Alexander Vardy. Maximum-likelihood decoding of reed-solomon codes is NP-hard. In *SODA '05: Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 470–478, 2005. 19, 20

[14] K Jambunathan. On choice of connection-polynomials for LFSR-Based stream ciphers. In *Progress in Cryptology INDOCRYPT 2000*, volume 1977 of *Lecture Notes in Computer Science*, pages 149–159. Springer Berlin / Heidelberg, 2000. 41

[15] Thomas Johansson and Fredrik Jnsson. Improved fast correlation attacks on stream ciphers via convolutional codes. pages 347–362. Springer-Verlag, 1999. 42

[16] A. K. Lenstra, H. W. Lenstra, Jr., and L. Lovász. Factoring polynomials with rational coefficients. *Math. Ann.*, 261(4):515–534, 1982. 10, 24, 34, 36

[17] H. W. Lenstra, Jr. Finding small degree factors of lacunary polynomials. In *Number theory in progress, Vol. 1 (Zakopane-Kościelisko, 1997)*, pages 267–276. de Gruyter, Berlin, 1999. 2, 32

[18] Subhamoy Maitra, Kishan Chand Gupta, and Ayineedi Venkateswarlu. Results on multiples of primitive polynomials and their products over gf(2). *Theoretical Computer Science*, 341(1-3):311 – 343, 2005. 41

[19] A. R. Meijer. Groups, factoring, and cryptography. *Math. Mag.*, 69(2):103–109, 1996. 37

[20] Thomas Muir. *A Treatise on the Theory of Determinants.* Dover Publications, 2003. 23

[21] Lawrence J. Risman. On the order and degree of solutions to pure equations. *Proc. Amer. Math. Soc.*, 55(2):261–266, 1976. 24

[22] J. B. Rosser and L. Schoenfeld. Approximate formulas for some functions of prime numbers. *Ill. J. Math.*, 6:64–94, 1962. 24, 25, 28

[23] Arne Storjohann. *Algorithms for Matrix Canonical Forms.* PhD thesis, Swiss Federal Institute of Technology Zürich, 2000. 6

[24] Alexander Vardy. The intractability of computing the minimum distance of a code. *IEEE Transactions on Information Theory*, 43(6):1757–1766, 1997. 1, 21