On Representations of Algebraic-Geometric Codes for List Decoding

Venkatesan Guruswami[1] and Madhu Sudan[2*]

[1] Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139. venkat@theory.lcs.mit.edu
[2] Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139. madhu@mit.edu

Abstract. We show that all algebraic-geometric codes possess a succinct representation that allows for the list decoding algorithms of [15, 7] to run in polynomial time. We do this by presenting a root-finding algorithm for univariate polynomials over function fields when their coefficients lie in finite-dimensional linear spaces, and proving that there is a polynomial size representation given which the root finding algorithm runs in polynomial time.

1 Introduction

The *list decoding* problem for an error-correcting code is that of reconstructing a list of all codewords within a specified Hamming distance from a received word. List decoding, which was introduced independently by Elias and Wozencraft [4, 18], offers a potential for recovery from errors beyond the traditional "error-correction radius" (i.e., half the minimum distance) of a code.

There has recently been a spurt of activity in the design of *efficient* list decoding algorithms for recovery from very high noise for several families of algebraic error-correcting codes [17, 15, 7, 8]. In particular, Shokrollahi and Wasserman [15] gave a list decoding algorithm for algebraic-geometric codes. Subsequently the authors [7] presented improvements to their algorithm, and increased further the bound on the number of errors that could be efficiently recovered from. Neither result is proven, however, to work unconditionally in polynomial time for all algebraic-geometric codes. Rather, the results are conditioned upon the existence of an efficient root-finding algorithm over the underlying algebraic function field. In turn, efficiency of known root-finding algorithms are dependent upon the size of the representation of certain basis elements of the underlying function field. Finally, no sufficiently succinct representations were known for the basis elements in the standard representation.

In this note, we examine and resolve this issue. We show that there exists a (non-standard) representation of the "basis" elements of a function field that is succinct and at the same time allows for efficient root-finding. As a consequence

* Supported in part by an MIT-NEC Research Initiation Award, a Sloan Foundation Fellowship and NSF Career Award CCR-9875511.

we show that every algebraic-geometric code has a succinct (polynomial-sized) representation such that the algorithms of [15, 7] run in polynomial time.

Our result shares an interesting role in the broader issue of "representation of codes" versus "complexity of decoding algorithms". It is well-known that the representation of an error-correcting code can have crucial impact on its decoding complexity. Some representations of a code allow for efficient decoding, while other representations of the same code may not yield an equally efficient algorithm for decoding. In fact this non-trivial feature laid the foundation for a public-key cryptosystem proposed by McEliece [11]. On the other hand some error-correcting codes are inherently hard to decode. This was established by Bruck and Naor [3] who proved that there are codes which are hard to decode even with unlimited preprocessing time and hence independent of representation. Our result shows that algebraic-geometric codes (henceforth AG-codes) do not belong to this class, i.e., they admit a polynomial size representation given which they can be list decoded in polynomial time up to a large radius. This representation may itself be hard to find; we do not address the complexity of finding it.

Our result may also be viewed as a parallel to results of Pellikaan [12] and Kotter [9] who proved, using the elegant idea of *error-correcting pairs*, that a succinct representation of AG-codes exists for the purpose of unambiguous decoding (up to essentially half the minimum distance). In this work we show that a similar result holds for list decoding as well.

We now give further details of the exact question we address. Existing list decoding algorithms for AG-codes [15, 7] run in polynomial time by assuming that operations over the underlying function field can be done efficiently, and in this work we would like to prove that the algorithm runs in polynomial time given a certain polynomial size representation of the code, without *any further assumptions*. Current list decoding algorithms for AG-codes (see [15, 7]) involve a two-step process. The first step is an "interpolation" step where one finds a (univariate) polynomial over the function field K that "fits" the received word. In the second step all roots of this polynomial in a B certain linear subspace of K are determined using a root-finding algorithm and these include all the candidate codewords within the list decoding radius.

The first of these steps works by exploiting linearity and solves the interpolation problem by finding a non-zero solution to a homogeneous linear system (the system is guaranteed to have a non-zero solution by a simple rank argument). The second step is more tricky, and Gao and Shokrollahi [5] recently gave a (randomized) polynomial time algorithm for root finding of univariate polynomials over function fields of nonsingular plane curves. Their algorithm would actually work over any function field modulo a specific representation of the function, in time polynomial in the size of their representation. However, the representation is not known to be succinct for all AG-codes. For a specific class of AG-codes, namely Hermitian Codes, [13] shows how to implement both steps, and in particular the root-finding step, efficiently. Wu and Siegel [19] present a different root-finding algorithm but which still assumes efficient operations over the underlying function field. Root-finding algorithms in [1] also assume efficient operations over the function field, and do not address if this can *always* be achieved through a succinct representation of the function field. In this work, we present a root-finding algorithm for general function fields which in fact uses the same approach as the one in [5]; our main contribution is a proof that the representation of the function field required by this algorithm is in fact terse, i.e., has size bounded by a polynomial in the relevant parameters. This in turn implies that there is a polynomial size representation of AG-codes given which they can be efficiently list decoded up to the radius bound achieved by [7].

Organization: We describe the basic notions about algebraic function fields that we will need in Section 2. In Section 3, we give a high level "algebraic" description of the list decoding algorithm of [7], and the root-finding algorithm that we will use, and then address the representation issues that these algorithms motivate. Section 4 describes the root finding algorithm formally for the representation we choose in Section 3. Finally, in Section 5, we give a formal description of the list decoding algorithm of [7] and state our main theorem that AG-codes have a succinct representation given which they can be efficiently list decoded.

2 Algebraic-geometric codes: Preliminaries

We now abstract the main notions associated with the theory of algebraic function fields that will be important to us. The interested reader may find further details in [16, 6]. In what follows we assume familiarity with the basic notions of field extensions.

An extension field K of a field k, denoted K/k, is an *algebraic function field* over k if the following conditions are satisfied: (i) There is an element $x \in K$ that is transcendental over k such that K is a finite extension of $k(x)$, and (ii) k is algebraically closed in K (i.e., the only elements in K that are algebraic over k are those in k).

For our applications to AG-codes, we will be interested in the case when k is a finite field, i.e., $k = \mathbb{F}_q$ for some prime power q. Thus $K = \mathbb{F}_q(X)[y_1, y_2, \ldots, y_m]$ where each y_i satisfies some polynomial equation over $\mathbb{F}_q(X)[y_1, \ldots, y_{i-1}]$. For the rest of this section, K will denote the function field in question.

Places, Valuations, and Degrees: A function field $K/\mathbb{F}_q$ has a set of *places* $\mathbb{P}_K$ and the associated set of *valuations*, given by a valuation map $v : \mathbb{P}_K \times K \to \mathbb{Z} \cup \{\infty\}$. The exact definition of these notions can be found, for instance, in [16]; we only abstract some properties relevant to us below.

Intuitively the places correspond to "points" on the algebraic curve associated with the function field K, and the valuation map tells us how many poles or zeroes a function in K has at a specific place in $\mathbb{P}_K$. It has the property that for any $f \in K$, there are only finitely many places $P \in \mathbb{P}_K$ such that $v(P, f) \neq 0$. As is normal practice, for each $P \in \mathbb{P}_K$, we denote by $v_P : K \to \mathbb{Z} \cup \{\infty\}$, the map $v_P(\cdot) = v(P, \cdot)$ which tells how many zeroes or poles a given function has at P (with the convention $v_P(0) = \infty$). The valuation v_P at any place satisfies the following properties:

(a) $v_P(a) = \infty$ iff $a = 0$ and $v_P(a) = 0$ for all $a \in \mathbb{F}_q \setminus \{0\}$.
(b) $v_P(ab) = v_P(a) + v_P(b)$ for all $a, b \in K \setminus \{0\}$.
(c) $v_P(a + b) \geq \min\{v_P(a), v_P(b)\}$.

Associated with every place is a *degree* abstracted via the map $\deg : \mathbb{P}_K \to \mathbb{Z}^+$. The degree, $\deg(P)$, of any place P is a positive integer and intuitively means the following: when we pick a function $f \in K$ which has no poles at P and "evaluate" it at P, we get a value in the field $\mathbb{F}_{q^{\deg(P)}}$. Thus places of degree one correspond to *rational points* on the curve.

Evaluations of functions at places: We can abstract the notion of *evaluation* of elements of the function field at the places by a map $\mathsf{eval} : K \times \mathbb{P}_K \to \bar{\mathbb{F}}_q \cup \{\infty\}$ (here $\bar{\mathbb{F}}_q$ is the algebraic closure of $\mathbb{F}_q$). This map has the following properties:

(i) $\mathsf{eval}(f, P) = \infty$ iff $v_P(f) < 0$, and $\mathsf{eval}(f, P) = 0$ iff $v_P(f) > 0$ for every $P \in \mathbb{P}_K$ and $f \in K$.
(ii) If $f \in K$, $P \in \mathbb{P}_K$ and $v_P(f) \geq 0$, then $\mathsf{eval}(f, P) \in \mathbb{F}_{q^{\deg(P)}}$.
(iii) The map eval respects field operations; i.e., if $v_P(f_1) \geq 0$ and $v_P(f_2) \geq 0$, then $\mathsf{eval}(f_1 + f_2, P) = \mathsf{eval}(f_1, P) + \mathsf{eval}(f_2, P)$, and $\mathsf{eval}(f_1 * f_2, P) = \mathsf{eval}(f_1, P) * \mathsf{eval}(f_2, P)$.

Divisors: The *divisor group* $\mathcal{D}_K$ of the function field K is a free abelian group on $\mathbb{P}_K$. An element D of $\mathcal{D}_K$ is thus represented by the formal sum $\sum_{P \in \mathbb{P}_K} a_P P$ where $a_P = 0$ for all but finitely many P; we say $D \succeq 0$ if $a_P \geq 0$ for all $P \in \mathbb{P}_K$. The support of a divisor D, denoted $\mathrm{supp}(D)$, is the (finite) set $\{P \in \mathbb{P}_K : a_P \neq 0\}$. The degree map extends naturally to a homomorphism $\deg : \mathcal{D}_K \to \mathbb{Z}$ as $\deg(\sum_P a_P P) = \sum_P a_P \deg(P)$.

For every $f \in K \setminus \{0\}$, there is an associated divisor, called the principal divisor and denoted (f), which is defined by $(f) = \sum_P v_P(f) P$. The following result is well known, and just states that every non-zero function in the function field has an equal number of zeroes and poles.

Theorem 1. *For any function field $K/\mathbb{F}_q$ and any $f \in K \setminus \{0\}$, $\deg((f)) = 0$.*

For every divisor $D \in \mathcal{D}_K$, one can define the linear space of functions $\mathcal{L}(D)$ as

$$\mathcal{L}(D) = \{f \in K : (f) + D \succeq 0\}.$$

For example for a divisor $D = aQ - bP$ where $P, Q \in \mathbb{P}_K$ and $a, b > 0$, $\mathcal{L}(D)$ is the space of all functions that have at least b zeroes at P and at most a poles at Q. It is known that for any divisor $D \succeq 0$, $\mathcal{L}(D)$ is a finite-dimensional vector space over $\mathbb{F}_q$ and $\dim(\mathcal{L}(D)) \leq 1 + \deg(D)$ (see [16] for a proof). A lower bound on $\dim(\mathcal{L}(D))$ is given by the celebrated Riemann-Roch theorem for function fields:

Theorem 2. [Riemann-Roch]: *Let $K/\mathbb{F}_q$ be any function field. There is a non-negative integer g, called the* genus *of $K/\mathbb{F}_q$, such that*

(a) For any divisor $D \in \mathcal{D}_K$, $\dim(\mathcal{L}(D)) \geq \deg(D) - g + 1$.
(b) There is an integer c, depending only on $K/\mathbb{F}_q$, such that $\dim(\mathcal{L}(D)) = \deg(D) - g + 1$ whenever $\deg(D) \geq c$. (In fact $c \leq 2g - 1$.)

Algebraic-geometric codes: We are now ready to define the notion of an AG-code (also known as *geometric Goppa code*). Let $K/\mathbb{F}_q$ be an algebraic function field of genus g, let $Q, P_1, P_2, \ldots, P_n$ be *distinct* places of degree one in $\mathbb{P}_K$, and let $G = P_1 + P_2 + \cdots + P_n$ and $D = \alpha Q$ be divisors of $K/\mathbb{F}_q$ (note that $\mathrm{supp}(G) \cap \mathrm{supp}(D) = \emptyset$).

The algebraic-geometric code $\mathcal{C}_{\mathcal{L}}(G, D) = \mathcal{C}_{\mathcal{L}}(G, \alpha, Q)$ is defined by[1]

$$\mathcal{C}_{\mathcal{L}}(G, \alpha, Q) := \{(\mathsf{eval}(f, P_1), \ldots, \mathsf{eval}(f, P_n)) : f \in \mathcal{L}(\alpha Q)\} \subseteq \mathbb{F}_q^n \ .$$

(Note that $\mathsf{eval}(f, P_i) \in \mathbb{F}_q$ since $v_{P_i}(f) \geq 0$ and $\deg(P_i) = 1$.) The following Proposition follows from the Riemann-Roch theorem and quantifies the parameters of these codes.

Proposition 3 *Suppose that $\alpha < n$. Then $\mathcal{C}_{\mathcal{L}}(G, \alpha, Q)$ is an $[n, k, d]_q$ code with $k = \dim(\mathcal{L}(\alpha Q)) \geq \alpha - g + 1$ and $d \geq n - \alpha$ (thus $k + d \geq n + 1 - g$).*[2]

3 Representation Issues

We now give a high level description of the list decoding algorithm of [7] and the root-finding algorithm over function fields that it uses. These will motivate the issues relating to representation of the function field that we address in this section.

The list decoding algorithm for an AG-code $\mathcal{C}_{\mathcal{L}}(G, \alpha, Q)$ over a function field $K/\mathbb{F}_q$ with $G = P_1 + P_2 + \cdots + P_n$ where $n = \deg(G)$, on input a received word $\boldsymbol{y} = (y_1, \ldots, y_n) \in \mathbb{F}_q^n$, works, at a high level, as follows:

Procedure LIST-DECODE$_{\mathcal{C}_{\mathcal{L}}(G,\alpha,Q)}(\boldsymbol{y}, e)$

Input: $\boldsymbol{y} \in \mathbb{F}_q^n$; error bound e.

Output: All elements $h \in \mathcal{L}(\alpha Q)$ such that $\mathsf{eval}(h, P_i) \neq y_i$ for at most e values of $i \in \{1, \ldots, n\}$.

1. "Fit" the pairs (y_i, P_i) by a "suitable" polynomial $H \in K[T]$. Specifically pick parameters ℓ, r, s appropriately (the exact choice will be specified in the full description of Section 5 and is not necessary for understanding the algorithm) and find a *non-zero* $H = \sum_{j=0}^{s} a_j T^j$ with $a_j \in \mathcal{L}((\ell - j\alpha)Q)$, $0 \leq j \leq s$, such that for $1 \leq i \leq n$, $H(h) \in \mathcal{L}(-rP_i)$ whenever $\mathsf{eval}(h, P_i) = y_i$.

[1] It is clear that the defined space is a linear space.
[2] The notation $[n, k, d]_q$ code stands, as usual, for a code over $\mathbb{F}_q$ of blocklength n, rate k and minimum distance d.

2. Find all roots $h \in \mathcal{L}(\alpha Q)$ of H and for each of them check if $\mathsf{eval}(h, P_i) \neq y_i$ for at most e values of i, and if so, output h.

Rationale behind the approach: Note that if $\mathsf{eval}(h, P_i) = y_i$ for at least $n - e$ values of i, then $H(h)$ has at least $r(n-e)$ zeroes (among the points $P_1, P_2, \ldots, P_n$ counting multiplicities). Also, by the choice of H, if $h \in \mathcal{L}(\alpha Q)$, we have $H(h) \in \mathcal{L}(\ell Q)$, and thus $H(h)$ has at most ℓ poles. Thus if $r(n-e) > \ell$, then $H(h) = 0$, or in other words h is a root of $H \in K[T]$. The parameters r, ℓ are chosen so that the algorithm is able to correct a "large" number of errors e.

The root-finding step referred to above can be implemented by the following algebraic procedure which finds all roots (in K) of a polynomial over K whose coefficients are restricted to lie in a linear space $\mathcal{L}(D)$ for some divisor $D \in \mathcal{D}_K$.

Procedure ROOT-FIND$_{(K,D)}(H)$

Input: A degree m polynomial $H = \sum_{i=0}^{m} a_i T^i \in K[T]$ where each $a_i \in \mathcal{L}(D)$.

Output: All roots of H that lie in K.

1. Find a divisor D' that depends on D such that all roots of H lie in $\mathcal{L}(D')$.
2. "Reduce" H modulo a place $R \in \mathbb{P}_K$ of large enough degree, say r (i.e., compute $b_i = \mathsf{eval}(a_i, R)$ for $0 \leq i \leq m$ and consider the polynomial $P = \sum_{i=0}^{m} b_i Y^i \in \mathbb{F}_{q^r}[Y]$).
3. Compute the roots, say $\alpha_1, \ldots, \alpha_t$, of P that lie in $\mathbb{F}_{q^r}$ using a root-finding algorithm for finite fields.
4. For each α_j, $1 \leq j \leq t$, "find" $\beta_i \in \mathcal{L}(D')$ such that $\mathsf{eval}(\beta_i, R) = \alpha_i$.

The above algebraic procedures raise several questions on how to represent the various objects associated with a function field, so as to be able to perform the associated computations efficiently. Since K is infinite, we will not try to represent *all* elements of K. Rather, we will always focus attention only on elements of K from the finite-dimensional space $\mathcal{L}(D)$. For any $D \succeq 0$, we can represent elements of $\mathcal{L}(D)$ as vectors in $\mathbb{F}_q^{\dim(\mathcal{L}(D))}$ which represent their coordinates with respect to some fixed basis of $\mathcal{L}(D)$ over $\mathbb{F}_q$. Since $\dim(\mathcal{L}(D)) \leq \deg(D) + 1$, this representation will be small provided $\deg(D)$ is not too large. Indeed, Step 1 of the decoding algorithm above is handled in [7] using such ideas.

The more tricky issue is in the root-finding step where we need to "evaluate" an $f \in \mathcal{L}(D)$ at some place $R \in \mathbb{P}_K$ of interest. To aid this we "represent" a place R by the values $\mathsf{eval}(\phi_i, R)$, for $1 \leq i \leq p$ where $p = \dim(\mathcal{L}(D))$ and $\phi_1, \ldots, \phi_p$ is a basis of $\mathcal{L}(D)$ over $\mathbb{F}_q$. Together with the representation of any element of $\mathcal{L}(D)$ as a linear combination of the ϕ_i's this clearly enables us to evaluate any element of $\mathcal{L}(D)$ at R. Since each $\mathsf{eval}(\phi_i, R) \in \mathbb{F}_{q^r}$ where $r = \deg(R)$, one can "represent" R, for purposes of evaluation by members of $\mathcal{L}(D)$, as an element of $\mathbb{F}_{q^r}^p$. (We assume that some standard representation of elements of $\mathbb{F}_{q^r}$.)

It should be somewhat clear that given these representations, the algebraic procedures discussed at the beginning of this section can in fact be turned into efficient algorithms. The next couple of sections prove formally that this is indeed the case.

4 Efficient Root-finding

In this section we consider the problem of finding the roots of a univariate polynomial $H \in K[T]$. We are interested in this task since it forms the critical component in list decoding algorithms for AG-codes. Considering the application to list decoding in mind, we only need to solve special instances of the univariate root-finding problem. Namely, we assume the input polynomial $H \in K[T]$ of degree m has all its coefficients in a linear subspace $\mathcal{L}(D)$ for some divisor $D \succeq 0$ of K, and the coefficients are input in the form discussed in the previous section. (For applications to AG-codes, the divisor D is actually a *one-point* divisor, i.e., is of the form ℓQ for some place Q of degree one, and moreover we will only be interested in roots that lie in $\mathcal{L}(\alpha Q)$ for some $\alpha \le \ell$. We, however, present a root-finding algorithm that works for any divisor, as this could be of independent interest.) In addition to this "uniform" input H, the algorithm also uses a non-uniform input, namely a place R of large degree, which only depends on D and does not depend on the degree of the input polynomial.

Before we specify the algorithm formally, we recall the high level description outlined in Section 3. The root-finding algorithm, on input a polynomial $H = \sum_{i=0}^{m} a_i T^i \in K[T]$ where each $a_i \in \mathcal{L}(D)$, first reduces H modulo R by evaluating each a_i at R. This gives a polynomial P of degree m over $\mathbb{F}_{q^r}$ (r is the degree of R). The algorithm then finds the roots in $\mathbb{F}_{q^r}$ of P, and then "lifts" these roots to give corresponding elements in K which are roots of H.

Algorithm ROOT-FIND$_{(K,D)}(H)$

Non-uniform input: (This depends only on D and is independent of the actual input) A place $R \in \mathbb{P}_K$ such that $\deg(R) = r > \deg(D)|\mathrm{supp}(D)|$ (such a place necessarily exists; see Lemma 1). The place R is represented as an s-tuple $(\zeta_1^R, \ldots, \zeta_s^R)$ over $\mathbb{F}_{q^r}$ comprising of evaluations of the basis functions $\mathcal{B}' = \{\phi_1, \phi_2, \ldots, \phi_p, \phi_{p+1}, \ldots, \phi_s\}$ of $\mathcal{L}(D')$ at R. Here $D' = \sum_{P \in \mathrm{supp}(D)} \frac{\deg(D)}{\deg(P)} P$, $s = \dim(\mathcal{L}(D'))$, and the basis $\mathcal{B}'$ *extends* a basis $\mathcal{B} = \{\phi_1, \ldots, \phi_p\}$ of $\mathcal{L}(D)$.

Input: A polynomial $H = a_0 + a_1 T + \cdots + a_m T^m$ of degree m in $K[T]$ where each $a_i \in \mathcal{L}(D)$ for some divisor D of K. Each a_i is presented as a p-tuple $(a_{i1}, \ldots, a_{ip})$ over $\mathbb{F}_q$ where $p = \dim(\mathcal{L}(D))$, $a_{ij} \in \mathbb{F}_q$ (this is to interpreted as $a_i = \sum_{j=1}^{p} a_{ij}\phi_j$ where $\mathcal{B} = \{\phi_1, \phi_2, \ldots, \phi_p\}$ is a basis of $\mathcal{L}(D)$ over $\mathbb{F}_q$).

Output: All the roots of H that lie in K. (The roots can have poles only at places in $\mathrm{supp}(D)$ and moreover cannot have too many poles at these places; in fact they all lie in $\mathcal{L}(D')$; see Lemma 2.)

Step 1: Reduce H modulo R to obtain a polynomial $h \in \mathbb{F}_{q^r}[T]$: i.e., for $0 \le i \le m$, compute

$$b_i = \mathsf{eval}(a_i, R) = \sum_{j=1}^{p} a_{ij}\zeta_j^R \in \mathbb{F}_{q^r},$$

and set $h[T] = b_0 + b_1 T + \cdots + b_m T^m$.

Step 2: Find all the (distinct) roots $\alpha_1, \alpha_2, \ldots, \alpha_t$ of h that lie in $\mathbb{F}_{q^r}$ using a standard root finding algorithm for finite fields. (This can be accomplished in deterministic $\text{poly}(q, r)$ time by an algorithm due to Berlekamp [2].)

Step 3 (Recovering the original roots): For each $\alpha_i \in \mathbb{F}_{q^r}$ such that $h(\alpha_i) = 0$ "find" the *unique* $\beta_i \in \mathcal{L}(D')$ such that $\mathsf{eval}(\beta_i, R) = \alpha_i$, in terms of its coefficients $c_{ij} \in \mathbb{F}_q$ with respect to the basis $\mathcal{B}'$ of $\mathcal{L}(D')$. (Recall that $D' = \sum_{P \in \text{supp}(D)} \frac{\deg(D)}{\deg(P)} P$ and thus $\deg(D') = \deg(D)|\text{supp}(D)|$, so Lemma 3 proves that such a β_i, if any, is unique.) For each i, the c_{ij}'s can be found by solving $\sum_{j=1}^{s} c_{ij} \zeta_j^R = \alpha_i$ which can be viewed as a linear system of equations over $\mathbb{F}_q$ (by fixing some representation of elements of $\mathbb{F}_{q^r}$ over $\mathbb{F}_q$).

Step 4: Output the list of roots $\{\beta_1, \ldots, \beta_t\}$.

It is clear, given the Lemmas referred to in the algorithm, that the algorithm correctly finds all roots of H in K. Moreover, it clearly runs in polynomial time given the non-uniform input. We thus get our main theorem of this section:

Theorem 4. *There is an efficient root-finding algorithm that, for any function field K and any divisor $D \succeq 0$, given an "advice" that depends only on D and is of size polynomial in $\deg(D)$, finds, in $\text{poly}(m, \deg(D))$ time, the roots of any degree m polynomial in $K[T]$ whose coefficients all lie in $\mathcal{L}(D)$.*

Lemma 1. *For any function field K, there exists a place of degree m in $\mathbb{P}_K$ for every large enough integer m.*

Proof: By the Hasse-Weil inequality, the number N_m of places of degree m in $\mathbb{P}_K$ satisfies $|N_m - q^m - 1| \leq 2gq^{m/2}$ where g is the genus of the function field K. Hence if $m \geq m_0$ where m_0 is the smallest integer that satisfies $\frac{q^{m_0}-1}{2q^{m_0/2}} > g$, then $N_m \geq 1$. □

Lemma 2. *If all coefficients of a non-zero polynomial $H \in K[T]$ lie in $\mathcal{L}(D)$ for some divisor $D \succeq 0$, then for all roots $\alpha \in K$ of H, $\alpha \in \mathcal{L}(D')$ where $D' = \sum_{P \in \text{supp}(D)} \frac{\deg(D)}{\deg(P)} P$. (Note that $\deg(D') = \deg(D)|\text{supp}(D)|$.)*

Proof: We first prove that if α has a pole at some place $P \in \mathbb{P}_K$, then $P \in \text{supp}(D)$. Indeed, suppose $H(\alpha) = 0$ and α has a pole at some $P \in \mathbb{P}_K \setminus \text{supp}(D)$. Since none of the coefficients of H have a pole at P, it follows that $H(\alpha)$ has a pole at P (in fact $v_P(H(\alpha)) \leq -m$ where m is the degree of H), and hence certainly $H(\alpha) \neq 0$, a contradiction.

We next prove that for $P \in \text{supp}(D)$, $v_P(\alpha) \geq -\frac{\deg(D)}{\deg(P)}$. Indeed, let $H[T] = a_m T^m + \ldots + a_1 T + a_0$ where each $a_j \in \mathcal{L}(D)$, and let $D = \sum_{R \in \text{supp}(D)} e_R R$ where each $e_R > 0$. Clearly $v_P(a_j \alpha^j) \geq -e_P + j v_P(\alpha)$ since $v_P(a_j) \geq -e_P$. If $v_P(\alpha) \geq 0$ we are done, so assume $v_P(\alpha) < 0$. Hence

$$v_P(H(\alpha) - a_m \alpha^m) = v_P(\sum_{j=0}^{m-1} a_j \alpha^j) \geq -e_P + (m-1) v_P(\alpha) \ . \qquad (1)$$

We now upper bound $v_P(a_m)$. Since $\deg((a_m)) = 0$ and $a_m \in \mathcal{L}(D)$, we have

$$\sum_{R \in \mathrm{supp}(D)} v_R(a_m) \deg(R) \leq 0,$$

and this together with $v_R(a_m) \geq -e_R$ for every R gives

$$v_P(a_m) \deg(P) \leq \sum_{R \in \mathrm{supp}(D) \setminus P} e_R \deg(R) = \deg(D) - e_P \deg(P) \ . \tag{2}$$

Thus we have

$$v_P(a_m \alpha^m) \leq \frac{\deg(D)}{\deg(P)} - e_P + m v_P(\alpha) \ . \tag{3}$$

Since $H(\alpha) = 0$, we must have $v_P(a_m \alpha^m) = v_P(H(\alpha) - a_m \alpha^m)$. Using Equations (1) and (3) this gives $-e_P + (m-1) v_P(\alpha) \leq \frac{\deg(D)}{\deg(P)} - e_P + m v_P(\alpha)$ which gives $v_P(\alpha) \geq -\frac{\deg(D)}{\deg(P)}$, as desired. □

Lemma 3. *If* $f_1, f_2 \in \mathcal{L}(A)$ *for some divisor* $A \succeq 0$ *and* $\mathsf{eval}(f_1, R) = \mathsf{eval}(f_2, R)$ *for some place* R *with* $\deg(R) > \deg(A)$*, then* $f_1 = f_2$.

Proof: Suppose not, so that $f_1 - f_2 \neq 0$. Then, by Theorem 1, $\deg((f_1 - f_2)) = 0$. But $f_1 - f_2 \in \mathcal{L}(A)$ and $v_R(f_1 - f_2) \geq 1$, so that $\deg((f_1 - f_2)) \geq \deg(R) - \deg(A) > 0$, a contradiction. Hence $f_1 = f_2$. □

5 An explicit list decoding algorithm

We now present an algorithm for list decoding AG-codes that runs in polynomial time given a (non-standard) representation of the code. The underlying algorithm is the same as the one in [7]; we refer the reader to [7] for details on the development of the algorithm, the specific choice of parameters and the correctness of the algorithm. We reproduce the algorithm here mainly to bring out the fact that the algorithm runs in polynomial time given the representation it assumes. The reader unfamiliar with [7] may find it useful to ignore the specific choice of parameters and just observe the parallel to the skeleton algorithm mentioned in Section 3.

Let $\mathcal{C} = \mathcal{C}_{\mathcal{L}}(G, \alpha, Q)$ be an AG-code where $G = P_1 + P_2 + \cdots + P_n$ and $Q \notin \mathrm{supp}(G)$. Let $\alpha < n$ and let g be the genus of K. Assume $\alpha > 2g - 2$ so that by the Riemann-Roch theorem $\dim(\mathcal{L}(\alpha Q)) = \alpha - g + 1$. Thus the rate k of $\mathcal{C}$ is $\alpha - g + 1$ and its designed distance is $d = n - \alpha$.

Algorithm LIST-DECODE$_{\mathcal{C}_{\mathcal{L}}(G,\alpha,Q)}(\boldsymbol{y}, e)$

Parameters: The algorithm computes and uses parameters r, ℓ (based on n, e, g).

Representation of $\mathcal{C}_{\mathcal{L}}(G, \alpha, Q)$: Fix a basis $\{\phi_{j_1} : 1 \leq j_1 \leq \ell - g + 1\}$ of $\mathcal{L}(\ell Q)$ such that $v_Q(\phi_{j_1}) \geq -(j_1 + g - 1)$ (i.e., ϕ_{j_1} has at most $j_1 + g - 1$ poles at Q). As shown in [7], for each i, $1 \leq i \leq n$, there exists a basis $\{\psi_{j_3, P_i} : 1 \leq j_3 \leq \ell - g + 1\}$ of $\mathcal{L}(\ell Q)$ such that $v_{P_i}(\psi_{j_3, P_i}) \geq j_3 - 1$. The explicit information which the decoding algorithm needs are the following:

(a) The values $\mathsf{eval}(\phi_{j_1}, P_i) \in \mathbb{F}_q$ for $1 \le i \le n$ and $1 \le j_1 \le \ell - g + 1$.
(b) The set $\{\alpha_{P_i,j_1,j_3} \in \mathbb{F}_q : 1 \le i \le n, 1 \le j_1, j_3 \le \ell - g + 1\}$ such that for every i and every j_1, we have $\phi_{j_1} = \sum_{j_3} \alpha_{P_i,j_1,j_3} \psi_{j_3,P_i}$ (as elements in K).
(c) A place $R \in \mathbb{P}_K$ with $\deg(R) = s > \ell$ represented through the evaluations of ϕ_{j_1} for $1 \le j_1 \le \ell - g + 1$ at R (these lie in $\mathbb{F}_{q^s}$).

Input: $\boldsymbol{y} = (y_1, y_2, \ldots, y_n) \in \mathbb{F}_q^n$; error bound e.
Output: A list of all codewords C in $\mathcal{C}_{\mathcal{L}}(G, \alpha, Q)$ such that $C_i \neq y_i$ for at most e values of i.

Step 0: Computer parameters r, ℓ such that

$$r(n-e) > \ell \text{ and } \frac{(\ell - g)(\ell - g + 1)}{2\alpha} > n\binom{r+1}{2}.$$

In particular set

$$r \stackrel{\text{def}}{=} 1 + \left\lfloor \frac{(2g+\alpha)n - 2ge + \sqrt{((2g+\alpha)n - 2ge)^2 - 4(g^2-1)((n-e)^2 - \alpha n)}}{2((n-e)^2 - \alpha n)} \right\rfloor,$$
$$\ell \stackrel{\text{def}}{=} r(n-e) - 1$$

Also set $s = \frac{\ell - g}{\alpha}$.

Step 1: Find $H \in \mathcal{L}(\ell Q)[T]$ of the form $H[T] = \sum_{j_2=0}^{s} \sum_{j_1=1}^{\ell-g+1-\alpha j_2} h_{j_1 j_2} \phi_{j_1} T^{j_2}$, i.e., find values of the coefficients $\{h_{j_1,j_2} \in \mathbb{F}_q\}$ such that the following conditions hold:
1. At least one h_{j_1,j_2} is non-zero.
2. For every $i \in [n]$, $\forall j_3, j_4$, $j_3 \ge 1$, $j_4 \ge 0$ such that $j_3 + j_4 \le r$,

$$h^{(i)}_{j_3,j_4} \stackrel{\text{def}}{=} \sum_{j_2=j_4}^{s} \sum_{j_1=1}^{\ell-g+1-\alpha j_2} \binom{j_2}{j_4} y_i^{j_2-j_4} \cdot h_{j_1,j_2} \alpha_{x_i,j_1,j_3} = 0.$$

/* This ensures that $\mathsf{eval}(h, P_i) = y_i$ implies $H(h) \in \mathcal{L}(-rP_i)$ for each i. */

Step 2: Using the root-finding algorithm described in Section 4 together with the place R which is supplied to the algorithm, "find" all roots $h \in \mathcal{L}(\alpha Q) \subseteq \mathcal{L}(\ell Q)$ of the polynomial $H \in K[T]$. For each such h, check if $\mathsf{eval}(h, P_i) = y_i$ for at least $(n-e)$ values of i, and if so, include h in output list. (Since h is "found" by finding its coefficients with respect to the basis functions ϕ_{j_1} and the algorithm is given the values $\mathsf{eval}(\phi_{j_1}, P_i)$ for $1 \le i \le n$ and $1 \le j_1 \le \ell - g + 1$, $\mathsf{eval}(h, P_i)$ can be computed efficiently.)
Step 3: Output the list of "codewords" h found in Step 2.

Since Step 1 just involves solving a homogeneous linear system of equations and Step 2 involves root-finding for which we gave an efficient algorithm in Section 4, it is clear that the above algorithm runs in polynomial time given the representation of the code it takes as input. We can thus state our main result:

Theorem 5 (Main theorem). *For every AG-code $\mathcal{C}_{\mathcal{L}}(G, \alpha, Q)$, there is a representation of the code of size polynomial in n (here $n = \deg(G)$), given which one can list decode from up to $n - \sqrt{n(n-d)}$ errors in polynomial time where $d = n - \alpha$ is the designed distance of the code.*

6 Conclusions

We have shown that AG-codes admit a representation given which the list decoding algorithm of [7] runs in polynomial time. It would be interesting to examine whether, for specific AG-codes which beat the Gilbert-Varshamov bound, say the codes based on the Garcia-Stichtenoth tower of function fields [6], this representation can also be *found* in polynomial time.

References

1. D. Augot and L. Pecquet. A Hensel lifting to replace factorization in list decoding of algebraic-geometric and Reed-Solomon codes. Manuscript, April 2000.
2. E. R. Berlekamp. Factoring polynomials over large finite fields. *Mathematics of Computations*, 24 (1970), pp. 713-735.
3. J. Bruck and M. Naor. The hardness of decoding linear codes with preprocessing. *IEEE Trans. on Information Theory*, Vol. 36, No. 2, March 1990.
4. P. Elias. List decoding for noisy channels. *Wescon Convention Record*, Part 2, Institute of Radio Engineers (now IEEE), pp. 94-104, 1957.
5. S. Gao and M. A. Shokrollahi. Computing roots of polynomials over function fields of curves. In *Proceedings of the Annapolis Conference on Number Theory, Coding Theory, and Cryptography*, 1999.
6. A. Garcia and H. Stichtenoth. Algebraic function fields over finite fields with many rational places. *IEEE Trans. on Info. Theory*, 41 (1995), pp. 1548-1563.
7. V. Guruswami and M. Sudan. Improved decoding of Reed-Solomon and Algebraic-geometric codes. *IEEE Trans. on Information Theory*, 45 (1999), pp. 1757-1767. Preliminary version appeared in *Proc. of FOCS'98.*
8. V. Guruswami and M. Sudan. List decoding algorithms for certain concatenated codes. *Proc. of STOC 2000*, to appear.
9. R. Kotter. A unified description of an error locating procedure for linear codes. *Proc. of Algebraic and Combinatorial Coding Theory*, 1992.
10. R. Matsumoto. *On the second step in the Guruswami-Sudan list decoding algorithm for AG-codes.* Technical Report of IEICE, pp. 65-70, 1999.
11. R. J. McEliece. A public-key cryptosystem based on algebraic coding theory. DSN Progress Report 42-44, Jet Propulsion Laboratory.
12. R. Pellikaan. On decoding linear codes by error correcting pairs. *Eindhoven Institute of Technology*, preprint, 1988.
13. R. R. Nielsen and T. Hoholdt. Decoding Hermitian codes with Sudan's algorithm. In *Proceedings of AAECC-13, LNCS 1719, Springer-Verlag*, 1999, pp. 260-270.
14. R. R. Nielsen and T. Hoholdt. Decoding Reed-Solomon codes beyond half the minimum distance. In *Coding Theory, Cryptograhy and Related areas*, (eds. Buchmann, Hoeholdt, Stichtenoth and H. tapia-Recillas) pp. 221-236, Springer 1999.

15. M. A. Shokrollahi and H. Wasserman. List decoding of algebraic-geometric codes. *IEEE Trans. on Information Theory*, Vol. 45, No. 2, March 1999, pp. 432-437.
16. H. Stichtenoth. *Algebraic Function Fields and Codes.* Springer-Verlag, Berlin, 1993.
17. M. Sudan. Decoding of Reed-Solomon codes beyond the error-correction bound. *Journal of Complexity*, 13(1):180-193, March 1997.
18. J. M. Wozencraft. List Decoding. *Quarterly Progress Report*, Research Laboratory of Electronics, MIT, Vol. 48 (1958), pp. 90-95.
19. Xin-Wen Wu and P. H. Siegel. Efficient list decoding of algebraic geometric codes beyond the error correction bound. In *Proc. of International Symposium on Information Theory*, June 2000, to appear.