Improved Decoding of Reed-Solomon and Algebraic-Geometric Codes

Venkatesan Guruswami* Madhu Sudan*

Abstract

Given an error-correcting code over strings of length n and an arbitrary input string also of length n, the list decoding problem is that of finding all codewords within a specified Hamming distance from the input string. We present an improved list decoding algorithm for decoding Reed-Solomon codes. The list decoding problem for Reed-Solomon codes reduces to the following "curve-fitting" problem over a field F: Given n points $\{(x_i.y_i)\}_{i=1}^n$, $x_i, y_i \in F$, and a degree parameter k and error parameter e, find all univariate polynomials p of degree at most k such that $y_i = p(x_i)$ for all but at most e values of $i \in \{1, \ldots, n\}$. We give an algorithm that solves this problem for $e < n - \sqrt{kn}$, which improves over the previous best result [22], for every *choice of k and n. Of particular interest is the case of $k/n > \frac{1}{3}$, where the result yields the first asymptotic improvement in four decades [15].*

The algorithm generalizes to solve the list decoding problem for other algebraic codes, specifically alternant codes (a class of codes including BCH codes) and algebraic-geometric codes. In both cases, we obtain a list decoding algorithm that corrects up to $n - \sqrt{n(n-d')}$ errors, where n is the block length and d' is the designed distance of the code. The improvement for the case of algebraic-geometric codes extends the methods of [19] and improves upon their bound for every choice of n and d'. We also present some other consequences of our algorithm including a solution to a weighted *curve fitting problem, which is of use in soft-decision decoding algorithms for Reed-Solomon codes.*

1 Introduction

An error correcting code $\mathcal{C}$ of block length n, rate k, and distance d over a q-ary alphabet Σ ($[n, k, d]_q$ code, for short) is a mapping from Σ^k (the message space) to Σ^n (the codeword space) such that any pair of strings in the range of $\mathcal{C}$ differ in at least d locations out of n. Reed-Solomon codes are a classical, and commonly used, construction of error-correcting codes that yield $[n, k+1, d = n-k]_q$ codes for any $k < n \leq q$. The alphabet Σ for such a code is a finite field F. The message specifies a polynomial of degree k over F in some formal variable x (by giving its $k+1$ coefficients). The mapping $\mathcal{C}$ maps this code to its evaluation at n distinct values of x chosen from F (hence it needs $q = |F| \geq n$). The distance property follows immediately from the fact that two degree k polynomials can agree in at most k places.

The decoding problem for an $[n, k, d]_q$ code is the problem of finding a codeword in Σ^n that is within a distance of e from a "received" word $R \in \Sigma^n$. In particular it is interesting to study the error-rate $\epsilon \stackrel{\text{def}}{=} e/n$ that can be corrected as a function of the message rate $\kappa \stackrel{\text{def}}{=} k/n$. For a family of Reed-Solomon codes of constant message rate and constant error rate, the two brute-force approaches to the decoding problem (compare with all codewords, or look at all words in the vicinity of the received word) take time exponential in n. It is therefore a non-trivial task to solve the decoding problem in polynomial time in n. Surprisingly, a classical algorithm due to Peterson [15] manages to solve this problem in polynomial time, as long as $e < \frac{n-k}{2}$ (i.e. achieves $\epsilon = (1-\kappa)/2$). Faster algorithms, with running time $O(n^2)$ or better, are also well-known: in particular the classical algorithms of Berlekamp and Massey (see [14] for a description) achieve such running time bounds. It is also easily seen that if $e \geq \frac{n-k}{2}$ then there may exist several different codewords within distance e of a received word, and so the decoding algorithm cannot possibly always recover the "correct" message if it outputs only one solution.

This motivates the list decoding problem, first defined in [6] (see also [7]) and sometimes also termed the bounded-distance decoding problem, that asks, given a received word $R \in \Sigma^n$, to reconstruct a list of all codewords within a distance e from the received word. List decoding offers a potential for recovery from errors beyond the traditional "error-correction" bound (i.e., the quantity $d/2$) of a code. Loosely, we refer to a list decoding algorithm reconstructing all codewords within distance e of a received word as an "e error-correcting" algorithm. Again we can study $\epsilon = e/n$ as a function of $\kappa = k/n$. Till recently, no asymptotic benefits were achieved using the list decoding approach to recover from errors. The only improvements known over the algorithm of [15] were decoding algorithms due to Sidelnikov [20] and Dumer [5] which correct

*Laboratory for Computer Science, MIT, 545 Technology Square, Cambridge, MA 02139, USA. email: {venkat,madhu}@theory.lcs.mit.edu.

$\frac{n-k}{2} + \Theta(\log n)$ errors, i.e., achieve $\epsilon = (1-\kappa)/2 + o(1)$. Recently, Sudan [22], building upon previous work of Ar et al. [1], presented a polynomial time list decoding algorithm for Reed-Solomon codes correcting at least $n - \sqrt{2kn}$ errors, thus achieving $\epsilon = 1 - \sqrt{2\kappa}$. (For an exact description of the number of errors corrected by this algorithm, see [23] or Figure 1.) A more efficient list decoding algorithm, running in time $O(n^2 \log^2 n)$, correcting the same number of errors has also been given by Roth and Ruckenstein [17]. For $\kappa \to 0$, this algorithm corrects an error rate $\epsilon \to 1$, thus allowing for nearly twice as many errors as the classical approach. For codes of rate greater than $1/3$, however, this algorithm does not improve over the algorithm of [15]. This case is of interest since applications in practice tend to use codes of high rates.

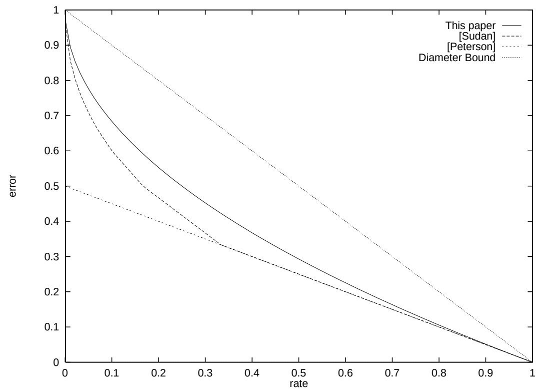

Figure 1. Error-correcting capacity plotted against the rate of the code for known algorithms.

In this paper we present a new polynomial-time algorithm for list-decoding of Reed-Solomon codes that corrects up to (exactly) $\left\lfloor n - \sqrt{nk} \right\rfloor$ errors (and thus achieves $\epsilon = 1 - \sqrt{\kappa}$). Thus our algorithm has a better error-correction rate than previous algorithms for every choice of $\kappa \in (0, 1)$; and in particular, for $\kappa > 1/3$ our result yields the *first asymptotic improvement* in the error-rate ϵ, since the original algorithm of [15]. (See Figure 1 for a graphical depiction of the relative error handled by our algorithm in comparison to previous ones.)

We solve the decoding problem by solving the following (more general) curve fitting problem: Given n pairs of elements $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ where $x_i, y_i \in F$, a degree parameter k and an error parameter e, find all univariate polynomials p such that $p(x_i) = y_i$ for at least $n - e$ values of $i \in \{1, \ldots, n\}$. Our algorithm solves this curve fitting problem for $e < n - \sqrt{nk}$. Our algorithm is based on the algorithm of [22] in that it uses properties of algebraic curves in the plane. The main modification is in the fact that we use the properties of "singularities" of these curves. As in the case of [22] our algorithm uses the notion of plane curves to reduce our problem to a bivariate polynomial factorization problem over F (actually only a root-finding problem for univariate polynomials over the rational function field $F(X)$). This task can be solved deterministically over finite fields in time polynomial in the size of the field or probabilistically in time polynomial in the logarithm of the size of the field and can also be solved deterministically over the rationals and reals [10, 12, 13]. Thus our algorithm ends up solving the curve-fitting problem over fairly general fields.

It is interesting to contrast our algorithm with results which show bounds on the number of codewords that may exist with a distance of e from a received word. One such result, due to Goldreich et al. [9], shows that the number of solutions to the list decoding problem is bounded by a polynomial in n if $e < n - \sqrt{n(n-d)}$. (A similar result has also been shown by Radhakrishnan [16].) Our algorithm proves this best known combinatorial bound "constructively" in that it produces a list of all such codewords in polynomial time. More recently, Justesen [11] has obtained upper bounds on the number of errors $e = e_{c,d,n}$ for which the output of a list decoding algorithm has at most c solutions for a constant c. The results of Justesen show that in the limit of large c, $e_{c,d,n}/n$ converges to $1 - \sqrt{1 - d/n}$ as we fix d/n and let $n \to \infty$. These bounds are of interest in that they hint at a potential limitation to further improvements to the list decoding approach.

Finally we point out that the main focus of this paper is on getting polynomial time algorithms maximizing the number of errors that may be corrected.

Extensions to Algebraic-Geometric Codes Algebraic-geometric codes are a class of algebraic codes that include the Reed-Solomon codes as a special case. These codes are of significant interest because they yield explicit construction of codes that beat the *Gilbert-Varshamov* bound over small alphabet sizes [24] (i.e., achieve higher value of d for infinitely many choices of n and k than that given by the probabilistic method). Decoding algorithms for algebraic-geometric codes are typically based on decoding algorithms for Reed-Solomon codes. In particular, Shokrollahi and Wasserman [19] generalize the algorithm of Sudan [22] for the case of algebraic-geometric codes. Specifically, they provide algorithms for factoring polynomials over some algebraic function fields; and then show how to decode using this factoring algorithm. Using a similar approach, we extend our decoding algorithm to the case of algebraic-geometric codes and obtain a list decoding algorithm correcting an $[n, k, d]_q$ algebraic-geometric code for up to $e < n - \sqrt{n(n-d)}$ errors, improving the previously known bound of $n - \sqrt{2n(n-d)} - g + 1$ errors (here g is the genus of the algebraic curve underlying the code). This algorithm uses a root-finding algorithm for univariate polynomials over algebraic function fields as a subroutine and some additional algorithmic assumptions about the underlying algebraic structures: The assumptions are described precisely in

Section 4.

Other extensions One aspect of interest with decoding algorithms is how they tackle a combination of erasures (i.e, some letters are explicitly lost in the transmission) and errors. Our algorithm generalizes naturally to this case. Another interesting extension of our algorithm is the solution to a *weighted* version of the curve-fitting problem[1]: Given a set of n pairs $\{(x_i, y_i)\}$ and associated non-negative integer weights $w_1, \ldots, w_n$, find all polynomials p such that $\sum_{i:p(x_i)=y_i} w_i > \sqrt{k \cdot \sum_{i=1}^{n} w_i^2}$. This generalization is of interest in "soft-decision" decoding of Reed-Solomon codes.

2 Reed-Solomon Decoding

We fix some notation first. In what follows F is a field and we will assume arithmetic over F to be of unit cost. $[n]$ will denote the set $\{1, \ldots, n\}$. For a vector $\vec{x} \in F^n$ and $i \in [n]$, the notation $\vec{x}_i$ will denote the ith coordinate of $\vec{x}$. $\Delta(\vec{x}, \vec{y})$ is the Hamming distance between strings $\vec{x}$ and $\vec{y}$, i.e., $|\{i|\vec{x}_i \neq \vec{y}_i\}|$.

Definition 1 (Reed-Solomon codes) *For parameters n, k and a field F of cardinality $n+1$, let α be a primitive nth root of unity in F (i.e., $\alpha^n = 1$ and $\alpha^i \neq 1$ for $i \in [n-1]$). The Reed-Solomon code with parameters n and k over the alphabet F with root α, denoted $C_{\mathrm{RS},F,\alpha,n,k}$, is the function mapping the messages F^{k+1} to code space F^n, given by $(C_{\mathrm{RS},F,\alpha,n,k}(\vec{m}))_j = \sum_{i=0}^{k} \vec{m}_{i+1}(\alpha^j)^i$, for $\vec{m} \in F^{k+1}$.*

Problem 1 (Reed-Solomon decoding)
INPUT: *String $\vec{y} \in F^n$; parameters k and e; and $\alpha \in F$.*
OUTPUT: *All messages $\vec{m} \in F^{k+1}$ such that $\Delta(C_{\mathrm{RS},F,\alpha,n,k}(\vec{m}), \vec{y}) \leq e$.*

Problem 2 (Polynomial reconstruction)
INPUT: *Integers k, t and n points $\{(x_i, y_i)\}_{i=1}^{n}$ where $x_i, y_i \in F$.*
OUTPUT: *All univariate polynomials p of degree at most k such that $y_i = p(x_i)$ for at least t values of $i \in [n]$.*
As pointed out earlier the Reed-Solomon code decoding problem reduces easily to the polynomial reconstruction problem.

2.1 Informal description of the algorithm

Our algorithm is based on the algorithm of [22], and so we review that algorithm first. The algorithm has two phases: In the first phase it finds a polynomial Q in two variables which "fits" the points (x_i, y_i), where fitting implies $Q(x_i, y_i) = 0$ for all $i \in [n]$. Then in the second phase it finds all *small degree roots* of Q i.e finds all polynomials p of degree at most k such that $Q(x, p(x)) \equiv 0$ or equivalently $y - p(x)$ is a *factor of* $Q(x, y)$; and these polynomials p form candidates for the output. The main assertions are that (1) if we allow Q to have a sufficiently large degree then the first phase will be successful in finding such a bivariate polynomial, and (2) if Q and p have low degree in comparison to the number of points where $y_i - p(x_i) = Q(x_i, y_i) = 0$, then $y - p(x)$ will be a factor of Q.

Our algorithm has a similar plan. We will find Q of low weighted degree that "fits" the points. But now we will expect more from the "fit". It will not suffice that $Q(x_i, y_i)$ is zero — we will require that every point (x_i, y_i) is a "singularity" of Q. Informally, a singularity is a point where the curve given by $Q(x, y) = 0$ intersects itself. We will make this notion formal as we go along. In our first phase the additional constraints will force us to raise the allowed degree of Q. However we gain (much more) in the second phase. In this phase we look for roots of Q and now we know that p passes through many singularities of Q, rather than just points on Q. In such a case we need only *half* as many singularities as regular points, and this is where our advantage comes from.

Pushing the idea further, we can force Q to intersect itself at each point (x_i, y_i) as many times as we want: in the algorithm described below, this will be a parameter r. There is no limit on what we can choose r to be: only our running time increases with r. We will choose r sufficiently large to handle as many errors as feasible. (In the weighted version of the curve fitting problem, we force the polynomial Q to pass through different points a different number r_i times, where r_i is proportional to the weight of the point.)

Finally, we come to the question of how to define "singularities". Traditionally, one uses the partial derivatives of Q to define the notion of a singularity. This definition is, however, not good for us since the partial derivatives over fields with small characteristic are not well-behaved. So we avoid this direction and define a singularity as follows: We first shift our coordinate system so that the point (x_i, y_i) is the origin. In the shifted world, we insist that all the monomials of Q with a non-zero coefficient be of sufficiently high degree. This will turn out to be the correct notion. (The algorithm of [22] can be viewed as a special case, where the coefficient of the constant term of the shifted polynomial is set to zero.)

We first define the shifting method precisely: For a polynomial $Q(x, y)$ and $\alpha, \beta \in F$ we will say that the shifted polynomial $Q_{\alpha,\beta}(x, y)$ is the polynomial given by $Q_{\alpha,\beta}(x, y) = Q(x + \alpha, y + \beta)$ Notice that shifting does not change the weighted degree for any weighting. Also, observe that following explicit relation between the coefficients $\{q_{ij}\}$ of Q and

[1] The evolution of the solution to the "curve-fitting" problem is somewhat interesting. The initial solutions of Peterson [15] did not explicitly solve the curve fitting problem at all. The solution provided by Welch and Berlekamp [27, 3] do work in this setting, even though the expositions there do not mention the curve fitting problem (see in particular, the description in [8]). Their problem statement, however, disallows repeated values of x_i. Sudan's [22] allows for repeated x_i's but does not allow for repeated pairs of (x_i, y_i). Our solution generalizes this one more step by allowing a weighting of (x_i, y_i)!

the coefficients $\{(q_{\alpha,\beta})_{ij}\}$ of $Q_{\alpha,\beta}$ holds:

$$(q_{\alpha,\beta})_{ij} = \sum_{i'\geq i}\sum_{j'\geq j}\binom{i'}{i}\binom{j'}{j} q_{i',j'}\alpha^{i'-i}\beta^{j'-j}.$$

In particular observe that the coefficients are obtained by a linear transformation of the original coefficients.

2.2 Algorithm

Definition 2 (weighted degree) *For non-negative weights w_1, w_2, the (w_1, w_2)-weighted degree of the monomial $x^i y^j$ is defined to be $iw_1 + jw_2$. For a bivariate polynomial $Q(x, y)$, and non-negative weights w_1, w_2, the (w_1, w_2)-weighted degree of Q, denoted (w_1, w_2)-wt-deg(Q), is the maximum over all monomials with non-zero coefficients in Q of the (w_1, w_2)-weighted degree of the monomial.*

We now describe our algorithm for the polynomial reconstruction problem.

- Input: $n, k, t, \{(x_i, y_i)\}_{i=1}^n$, where $x_i, y_i \in F$.
- Parameters: r, l:

$$r \stackrel{\text{def}}{=} 1 + \left\lfloor \frac{kn + \sqrt{k^2n^2 + 4(t^2 - kn)}}{2(t^2 - kn)} \right\rfloor,$$

and $l \stackrel{\text{def}}{=} rt - 1$.

Step 1: Find a polynomial $Q(x, y)$ such that $(1, k)$-wt-deg$(Q) \leq l$, i.e., find values for its coefficients $\{q_{j_1 j_2}\}_{j_1,j_2\geq 0: j_1+kj_2\leq l}$ such that the following conditions hold:

1. At least one q_{j_1,j_2} is non-zero
2. For every $i \in [n]$, if $Q^{(i)}$ is the shift of Q to (x_i, y_i), then all coefficients of $Q^{(i)}$ of total degree less than r are 0. More specifically:

$$\forall i \in [n], \forall\, j_1, j_2 \geq 0, \text{ s.t. } j_1 + j_2 < r,$$

$$q^{(i)}_{j_1 j_2} \stackrel{\text{def}}{=} \sum_{j_1'\geq j_1}\sum_{j_2'\geq j_2}\binom{j_1'}{j_1}\binom{j_2'}{j_2} q_{j_1',j_2'} x_i^{j_1'-j_1} y_i^{j_2'-j_2} = 0.$$

Step 2: Find all polynomials $p \in \mathcal{F}_q[X]$ of degree at most k such that p is a root of Q (i.e, $y - p(x)$ is a factor of $Q(x, y)$). For each such polynomial p check if $p(x_i) = y_i$ for at least t values of $i \in [n]$, and if so, include p in output list.

Remark: Step 2 above can be performed in polynomial time by using the bivariate polynomial factorization algorithm of Grigoriev [10] or Kaltofen [12] (that work over the finite fields as well as the rationals) or by using the more efficient root-finding algorithm of Shokrollahi [18] (for the finite field case).

2.3 Analysis of the Algorithm

We now proceed to prove our main result on polynomial reconstruction, stated as Theorem 7 at the end of this section. We will first prove a few necessary lemmas. In what follows Q can be any polynomial returned in Step 1 of the algorithm.

Lemma 3 *If (x_i, y_i) is an input point and p is any polynomial such that $y_i = p(x_i)$, then $(x - x_i)^r$ divides $g(x) \stackrel{\text{def}}{=} Q(x, p(x))$.*

Proof: Let $p'(x)$ be the polynomial given by $p'(x) = p(x + x_i) - y_i$. Notice that $p'(0) = 0$. Hence $p'(x) = xp''(x)$, for some polynomial $p''(x)$. Now, consider $g'(x) \stackrel{\text{def}}{=} Q^{(i)}(x, p'(x))$ We first argue that $g'(x - x_i) = g(x)$. To see this, observe that

$$g(x) = Q(x, p(x)) = Q^{(i)}(x - x_i, p(x) - y_i) =$$

$$Q^{(i)}(x - x_i, p'(x - x_i)) = g'(x - x_i).$$

Now, by construction, $Q^{(i)}$ has no coefficients of total degree less than r. Thus by substituting $y = xp''(x)$ for y, we are left with a polynomial g' such that x^r divides $g'(x)$. Shifting back we have $(x - x_i)^r$ divides $g'(x - x_i) = g(x)$. □

Lemma 4 *If $p(x)$ is a polynomial of degree at most k such that $y_i = p(x_i)$ for at least t values of $i \in [n]$ and $rt > l$, then $y - p(x)$ divides Q.*

Proof: Consider the polynomial $g(x) = Q(x, p(x))$. By the definition of weighted degree, and the fact that the $(1, k)$-weighted degree of Q is at most l, we have that g is a polynomial of degree at most l. By Lemma 3, for every i such that $y_i = p(x_i)$, we know that $(x - x_i)^r$ divides $g(x)$. Thus if S is the set of i such that $y_i = p(x_i)$, then $\prod_{i\in S}(x - x_i)^r$ divides $g(x)$. (Notice in particular that $x_i \neq x_j$ for any pair $i \neq j \in S$, since then we would have $(x_i, y_i) = (x_i, p(x_i)) = (x_j, p(x_j)) = (x_j, y_j)$.) By the hypothesis $|S| \geq t$, and hence we have a polynomial of degree at least rt dividing g which is a polynomial of degree at most $l < rt$. This can happen only if $g \equiv 0$. Thus we find that $p(x)$ is a root of $Q(x, y)$ (where the latter is viewed as a polynomial in y with coefficients from the ring of polynomials in x). By the division algorithm, this implies that $y - p(x)$ divides $Q(x, y)$. □

All that needs to be shown now is that a polynomial Q as sought for in Step 1 does exist. The lemma below shows this conditionally.

Lemma 5 *If $n\binom{r+1}{2} < \frac{l(l+2)}{2k}$, then a polynomial Q as sought in Step 1 does exist (and can be found in polynomial time by solving a linear system).*

Proof: Notice that the computational task in Step 1 is that of solving a homogeneous linear system. A non-trivial solution exists as long as the rank of the system is strictly smaller than the number of unknowns. The rank of the system may be bounded from above by the number of constraints, which is $n\binom{r+1}{2}$. The number of unknowns equals the number of monomials of $(1,k)$-weighted degree at most l and this number equals

$$\begin{aligned}\sum_{j_2=0}^{\lfloor \frac{l}{k}\rfloor}\sum_{j_1=0}^{l-kj_2} 1 = \sum_{j_2=0}^{\lfloor \frac{l}{k}\rfloor}(l+1-kj_2) \\ &= (l+1)\left(\left\lfloor\frac{l}{k}\right\rfloor+1\right)-\frac{k}{2}\left\lfloor\frac{l}{k}\right\rfloor\left(\left\lfloor\frac{l}{k}\right\rfloor+1\right) \\ &\geq \left(\left\lfloor\frac{l}{k}\right\rfloor+1\right)\left(l+1-\frac{l}{2}\right) \\ &\geq \frac{l}{k}\cdot\frac{l+2}{2},\end{aligned}$$

and the result follows. □

Lemma 6 *If n, k, t satisfy $t^2 > kn$, then for the choice of r, l made in our algorithm, $n\binom{r+1}{2} < \frac{l(l+2)}{2k}$ and $rt > l$ both hold.*

Proof: Since $l \stackrel{\text{def}}{=} rt-1$ in our algorithm, $rt > l$ certainly holds. Using $l = rt-1$, we now need to satisfy the constraint

$$n\binom{r+1}{2} < \frac{(rt-1)(rt+1)}{2k}$$

which simplifies to $r^2t^2 - 1 > kn(r^2+r)$ or, equivalently,

$$r^2(t^2-kn) - knr - 1 > 0.$$

Hence it suffices to pick r to be an integer greater than the larger root of the above quadratic, and therefore picking

$$r = 1 + \left\lfloor \frac{kn + \sqrt{k^2n^2 + 4(t^2-kn)}}{2(t^2-kn)} \right\rfloor$$

suffices, and this is exactly the choice made in the algorithm. □

Theorem 7 *The polynomial reconstruction problem problem can be solved in time polynomial in n, provided $t > \sqrt{kn}$, for every field F, using as a subroutine an algorithm for finding roots of a univariate polynomial over the rational function field $F(X)$.*

Proof: Follows from Lemma 4, Lemma 5, and Lemma 6. □

Corollary 8 *Given a family of Reed-Solomon codes of message rate κ, an error-rate of $\epsilon = 1-\sqrt{k}$ can be list-decoded in polynomial time.*

Remark: Since r affects the running time of our algorithm, we point out that if one is given $t^2 \geq (1+\epsilon)kn$, for some constant $\epsilon > 0$, then r may be set to some constant depending on ϵ *alone* (and independent of n, k, t), assuming the rate k/n of the code is a constant. In this case, therefore, the number of polynomials output will be at most a *constant* (that depends on ϵ).

3 Some easy extensions

We start by describing some easy consequences and extensions of the algorithm of Section 2. The first three results are just applications of the curve-fitting algorithm. The fourth result revisits the curve-fitting algorithm to get a solution to a weighted curve-fitting problem.

3.1 Alternant codes

We first describe a generalization of Reed Solomon codes termed alternant codes that includes a wide family of codes such as BCH codes, Goppa codes etc.

Definition 9 (Alternant Codes ([14], §12.2)) *For positive integers m, k_0, n, prime power q, vector $\vec{\alpha}$ of distinct elements $\alpha_1,\ldots,\alpha_n \in \mathrm{GF}(q^m)$, and vector $\vec{v}$ of nonzero elements $v_1,\ldots,v_n \in \mathrm{GF}(q^m)$, the Generalized Reed-Solomon code $\mathrm{GRS}_{k_0}(\vec{\alpha},\vec{v})$ is the code whose messages correspond to polynomials $p \in \mathrm{GF}(q^m)[x]$ of degree $< k_0$ and where the encoding of the message p is the string $(v_1p(\alpha_1),\ldots,v_np(\alpha_n))$. The alternant code $\mathcal{A}_{k_0}(\vec{\alpha},\vec{v})$ is the intersection of $\mathrm{GRS}_{k_0}(\vec{\alpha},\vec{v})$ with $\mathrm{GF}(q)^n$.*

It is obvious that the Generalized Reed-Solomon code has distance at least $d' = n - k_0 + 1$. Thus this holds also for the alternant code. We term this the *designed distance* of the alternant code. The actual rate and distance of the code are harder to determine. The rate lies somewhere between $n - m(n-k_0)$ and k_0 and thus the distance d lies between d' and md'. Playing with the vector $\vec{v}$ might alter the rate and the distance (which is presumably why it is used as a parameter).

The decoding algorithm of the previous section can be used to decode alternant codes as well. Given a received word $(r_1,\ldots,r_n) \in \mathrm{GF}(q)^n$, we use as input to the polynomial reconstruction problem the pairs $\{(x_i,y_i)\}_{i=1}^n$, where $x_i = \alpha_i$ and $y_i = r_i/v_i$ are elements of $\mathrm{GF}(q^m)$. The list of polynomials output includes all possible codewords from the alternant code. Thus the decoding algorithm for the earlier section is really a decoding algorithm for alternant codes as well; with the caveat that its performance can only be compared with the designed distance, rather than the actual distance. The following theorem summarizes the scope of the decoding algorithm.

Theorem 10 *Let $\mathcal{A}$ be an $[n, k, d]_q$ alternant code with designed distance d' (and thus satisfying $\frac{d}{m} \le d' \le d$). Then there exists a polynomial time list decoding algorithm for $\mathcal{A}$ decoding up to $e < n - \sqrt{n(n-d')}$ errors.*

(We note that decoding algorithms given in classical texts seem to correct $d'/2$ errors. We are unaware of the more recent work on this topic.)

3.2 Other models of error

The algorithm of Section 2 is also capable of dealing with other notions of corruption of information. A much weaker notion of corruption (than an "error") in data transmission is that of an "erasure": Here a transmitted symbol is either simply "lost" or received in obviously corrupted shape. We now note that the decoding algorithm of Section 2 handles the case of errors and erasure naturally. Suppose n symbols were transmitted and $n' \le n$ were received and s symbols got erased. (We stress that the problem definition specifies that the receiver knows which symbols are erased.) The problem just reduces to a polynomial reconstruction problem on n' points. An application of Theorem 7 yields that e errors can be corrected provided $e < n' - \sqrt{n'k}$. Thus we get:

Theorem 11 *The list-decoding problem for $[n, k+1, d]_q$ Reed-Solomon codes allowing for e errors and s erasures can be solved in polynomial time, provided $e + s < n - \sqrt{(n-s)k}$.*

The classical results of this nature show that one can solve the decoding problem if $2e + s < n - k$. To compare the two results we restate both result. The classical result can be rephrased as

$$n - (s + e) > \frac{n - s + k}{2},$$

while our result requires that

$$n - (s + e) > \sqrt{(n-s)k}.$$

By the AM-GM inequality it is clear that the second one holds whenever the first holds.

3.3 Decoding with uncertain receptions

Consider the situation when, instead of receiving a single word $y = y_1, y_2, \ldots, y_n$, for each $i \in [n]$ we receive a list of l possibilities $y_{i1}, y_{i2}, \ldots, y_{il}$ such that one of them is correct (but we do not know which one). Once again, as in normal list decoding, we wish to find out all possible codewords which could have been possibly transmitted, except that now the guarantee given to us is not in terms of the number of errors possible, but in terms of the maximum number of uncertain possibilities at each position of the received word. Let us call this problem *decoding from uncertain receptions*. One can prove, along the lines of the proof of Theorem 7, that:

Theorem 12 *List decoding from uncertain receptions on a $[n, k+1, d = n-k]_q$ Reed-Solomon code can be done in polynomial time provided the number of "uncertain possibilities" l at each position $i \in [n]$ is (strictly) less than n/k.*

3.4 Weighted curve fitting

Another natural extension of the algorithm of Section 2 is to the case of weighted curve fitting. This case is somewhat motivated by a decoding problem called the *soft-decision decoding* problem (see [26] for a formal description), as one might use the reliability information on the individual symbols in the received word more flexibly by encoding them appropriately as the weights below instead of declaring erasures. At this point we do not formalize the explicit connection between the two. Instead we just state the weighted curve fitting problem and describe our solution to this problem.

Problem 3 (Weighted polynomial reconstruction)
INPUT: *n points $\{(x_1, y_1), \ldots, (x_n, y_n)\}$, n non-negative integer weights $w_1, \ldots, w_n$, and parameters k and t.*
OUTPUT: *All polynomials p such that $\sum_{i:p(x_i)=y_i} w_i$ is at least t.*

The algorithm of Section 2 can be modified as follows: In Step 1, we could find a polynomial Q which has a singularity of order $w_i r$ at the point (x_i, y_i). Thus we would now have $\sum_{i=1}^{n} \binom{rw_i+1}{2}$ constraints. If a polynomial p passes through the points (x_i, y_i) for $i \in S$, then $y - p(x)$ will appear as a factor of $Q(x, y)$ provided $\sum_{i \in S} rw_i$ is greater than $(1, k)$-wt-deg(Q). Optimizing over the weighted degree of Q yields the following theorem.

Theorem 13 *The weighted polynomial reconstruction problem can be solved in time polynomial in the sum of w_i's provided $t > \sqrt{k \sum_{i=1}^{n} w_i^2}$.*

Remark: The fact that the algorithm runs in time pseudo-polynomial in w_i's should not be a serious problem. Given any vector of real weights, one can truncate and scale the w_i's without too much loss in the value of t for which the problem can be solved.

4 Algebraic-Geometric Codes

We now describe the extension of our algorithm to the case of algebraic-geometric codes. Much of the work involved here is mainly in ferreting out the axioms satisfied by these constructions. We do so in Section 4.1. The algorithm then follows along the lines of the algorithm of Shokrollahi and Wasserman [19], modulo some algorithmic assumptions about the underlying structures. (Such assumptions are inevitable: the class of algebraic geometric codes are vast and not all of them have constructive algorithms associated with them. Our assumptions are nearly the same as those of [19].) Our algorithm yields an algorithm for list decoding which corrects up to $e < n - \sqrt{n(n-d)}$ errors in an $[n,k,d]_q$ code, improving the result of [19], which corrects up to $e < n - \sqrt{2n(n-d)} - g + 1$ errors.

4.1 Definitions

An algebraic-geometric code is built over a structure termed an algebraic function field. An algebraic function field is described by a six-tuple $\mathcal{A} = (\mathcal{F}_q, \overline{\mathcal{X}}, \mathcal{X}, K, g, \mathsf{ord})$, where:

- $\mathcal{F}_q$ is a finite field with q elements, with $\overline{\mathcal{F}_q}$ denoting its algebraic closure.
- $\overline{\mathcal{X}}$ is a set of *points* (typically some subset of (variety in) $\overline{\mathcal{F}_q}^l$, but this will be irrelevant to us).
- $\mathcal{X}$ is a subset of $\overline{\mathcal{X}}$, called the *rational* points of $\overline{\mathcal{X}}$.
- K is a set of functions from $\overline{\mathcal{X}}$ to $\overline{\mathcal{F}_q} \cup \{\infty\}$ (where ∞ is a special symbol representing an undefined value). It is usually customary to refer to just K as the function field (and letting the other components of $\mathcal{A}$ be implicit).
- $\mathsf{ord} : K \times \overline{\mathcal{X}} \to \mathcal{Z}$. $\mathsf{ord}(f,x)$ is called the *order* of the function f at point x.
- g is a non-negative integer called the *genus* of $\mathcal{A}$.

The components of $\mathcal{A}$ always satisfy the following properties:

1. K is a field extension of $\mathcal{F}_q$: K is endowed with operations $+$ and $*$ giving it a field structure. Furthermore, for $f, g \in K$, the functions $f+g$ and $f*g$ satisfy $f(x)+g(x) = (f+g)(x)$ and $(f*g)(x) = f(x)*g(x)$, provided $f(x)$ and $g(x)$ are defined. Finally, corresponding to every $\alpha \in \mathcal{F}_q$, there exists a function $\alpha \in K$ s.t. $\alpha(x) = \alpha$ for every $\mathcal{X} \in \overline{\mathcal{X}}$. (In what follows we let αf denote the function $\alpha * f$.)

2. Rational points: For every $f \in K$ and $x \in \mathcal{X}$, $f(x) \in \mathcal{F}_q \cup \{\infty\}$.

3. Order properties: (The order is a common generalization of the degree of a function as well as its zeroes. Informally, the quantity $\sum_{x \in \overline{\mathcal{X}} : \mathsf{ord}(f,x) > 0} \mathsf{ord}(f,x)$ is analogous to the degree of a function. If $\mathsf{ord}(f,x) < 0$, then the negative of $\mathsf{ord}(f,x)$ is the number of zeroes f has at the point x. The following axioms may make a lot of sense when this is kept in mind.)
   For every $f, g \in K - \{0\}, \alpha, \beta \in \mathcal{F}_q, x \in \overline{\mathcal{X}}$: the order function ord satisfies:

   (a) $f(x) = 0 \iff \mathsf{ord}(f,x) < 0$; $f(x) = \infty \iff \mathsf{ord}(f,x) > 0$.

   (b) $\mathsf{ord}(f*g,x) = \mathsf{ord}(f,x) + \mathsf{ord}(g,x)$.

   (c) $\mathsf{ord}(\alpha f + \beta g, x) \leq \max\{\mathsf{ord}(f,x), \mathsf{ord}(g,x)\}$.

4. Distance property: If $\sum_{x \in \overline{\mathcal{X}}} \mathsf{ord}(f) < 0$, then $f \equiv 0$. (This property is just the generalization of the well-known theorem showing that a degree d polynomial may have at most d zeroes.)

5. Rate property: Observe that, by Property 3(c) above, the set of functions $\vee_{i,x} = \{f | \mathsf{ord}(f,x) \leq i\}$ form a vector space over $\mathcal{F}_q$, for any $x \in \overline{\mathcal{X}}$ and $i \in \mathcal{Z}$. Of particular interest will be functions which may have positive order at only one point $x_0 \in \overline{\mathcal{X}}$ and nowhere else. Let $L_{i,x}$ denote the set $\{f \in K | \mathsf{ord}(f,x) \leq i \wedge \mathsf{ord}(f,y) \leq 0, \forall y \in \overline{\mathcal{X}} - \{x\}\}$. Since $L_{i,x} = \vee_{i,x} \cap (\cap_{y \in \overline{\mathcal{X}} - \{x\}} \vee_{0,y})$, we have that L is also a vector space over $\mathcal{F}_q$. The rate property is that for every $i \in \mathcal{Z}$, $x \in \mathcal{X}$, $L_{i,x}$ is a vector space of dimension at least $i - g + 1$. (This property is obtained from the famed *Riemann-Roch theorem* for the actual realizations of $\mathcal{A}$, and in fact the dimension is exactly $i - g + 1$ if $i > 2g - 2$.)

The following lemma shows how to construct a code from an algebraic function field, given $n+1$ rational points.

Lemma 14 *If there exists an algebraic function field* $\mathcal{A} = (\overline{\mathcal{F}_q}, \overline{\mathcal{X}}, \mathcal{X}, K, g, \mathsf{ord})$ *with* $n+1$ *distinct rational points* $x_0; x_1, \ldots, x_n$, *then the linear space* $\mathcal{C} = \{(f(x_1), \ldots, f(x_n)) | f \in L_{k+g-1,x_0}\}$ *form an* $[n, k', d']_q$ *code for some* $k' \geq k$ *and* $d' \geq n - k - g + 1$.

Proof: For $i \geq 1$, by Property 2, we have that $f(x_i) \in \mathcal{F}_q \cup \{\infty\}$, and by Property 3a we have that $f(x_i) \neq \infty$. Thus $\mathcal{C} \subseteq \mathcal{F}_q^n$. By Property 4, the map $\mathsf{ev} : L_{k+g-1,x_0} \longrightarrow \mathcal{F}_q^n$ given by $f \mapsto (f(x_1), f(x_2), \ldots, f(x_n))$ is one-one, and hence $\dim(\mathcal{C}) = \dim(L_{k+g-1,x_0})$. By Property 5, this implies $\mathcal{C}$ has dimension at least k, yielding $k' \geq k$. Finally, consider $f_1 \neq f_2 \in L_{k+g-1,x_0}$ that agree in $k+g$ places. If f_1 and f_2 agree at x_i, then $(f_1 - f_2)(x_i) = 0$ and thus by Property 3a, $\mathsf{ord}(f_1 - f_2, x_i) < 0$. Furthermore, we have that for every $x \in \overline{\mathcal{X}} - \{x_0\}$, $\mathsf{ord}(f_1 - f_2, x) \leq 0$. Finally at x_0 we have $\mathsf{ord}(f_1 - f_2, x_0) \leq k + g - 1$. Thus summing over all $x \in \overline{\mathcal{X}}$, we have $\sum_{x \in \overline{\mathcal{X}}} \mathsf{ord}(f_1 - f_2, x) < 0$ and thus $f_1 - f_2 \equiv 0$ using Property 4 above. This yields the distance property as required. □

Codes constructed as above and achieving $d/n, k/n > 0$ (in the limit of large n) are known for constant alphabet size q. In fact, such codes achieving bounds better than those known by probabilistic constructions are known for $q \geq 49$ [24].

4.2 The Decoding Algorithm

We now describe the extension of our algorithm to the case of algebraic-geometric codes. As usual we will try to describe the data points $\{(x_i, y_i)\}$ by some polynomial Q.

We follow [19] and let Q be a polynomial in a formal variable y with coefficients from K (i.e., $Q[y] \in K[y]$). Now given a value of $y_i \in \mathcal{F}_q$, $Q[y_i]$ will yield an element of K. By definition such an element of K has a value at $x_i \in \mathcal{X}$ and just as in [19] we will also require $Q(x_i, y_i) = Q[y_i](x_i)$ to evaluate to zero. We, however, will require more and insist that (x_i, y_i) "behave" like a zero of multiplicity r of Q; since $x_i \in \mathcal{X}$ and $y_i \in \mathcal{F}_q$, we need to be careful in specifying the conditions to achieve this. We, as in [19], also insist that Q has a small (but positive) order l at x_0 for any substitution of y with a function in K of order at most $\alpha \stackrel{\rm def}{=} k + g - 1$ at the point x_0. Having found such a Q, we then look for *roots* $h \in K$ of Q.

What remains to be done is to explicitly express the conditions (i) (x_i, y_i) *behaves* like a zero of order r of Q for $1 \le i \le n$, and (ii) $\mathsf{ord}(Q[f], x_0) \le l$ for any $f \in L_{\alpha, x_0}$, where l is a parameter that will be set later (and which will play the same role as the l in our decoding algorithm for Reed-Solomon codes). To do so, we assume that we are explicitly given functions $\phi_1, \ldots, \phi_{l-g+1}$ such that $\mathsf{ord}(\phi_j, x_0) \le j + g - 1$ and such that $\mathsf{ord}(\phi_j, x_0) < \mathsf{ord}(\phi_{j+1}, x_0)$. Let $s \stackrel{\rm def}{=} \left\lfloor \frac{l-g}{\alpha} \right\rfloor$. We will then look for coefficients q_{j_1,j_2} such that

$$Q[y] = \sum_{j_2=0}^{s} \sum_{j_1=1}^{l-g+1-\alpha j_2} q_{j_1 j_2} \phi_{j_1} y^{j_2}. \qquad (1)$$

By explicitly setting up Q as above, we impose the constraint (ii) above. To get constraint (i) we need to "shift" our basis. This is done exactly as before with respect to y_i, however, $x_i \in \mathcal{X}$ and hence a different method is required to handle it. The following lemmas show how this may be achieved.

Lemma 15 *For every $f, g \in K$, and $x \in \mathcal{X}$, there exist $\alpha_0, \beta_0 \in \mathcal{F}_q$, $(\alpha_0, \beta_0) \ne (0,0)$ such that*

$$\mathsf{ord}(\alpha_0 f + \beta_0 g, x) < \max\{\mathsf{ord}(f,x), \mathsf{ord}(g,x)\}.$$

Proof: Notice that if $\mathsf{ord}(f,x) \ne \mathsf{ord}(g,x)$, then there is nothing to prove. So we assume $\mathsf{ord}(f,x) = \mathsf{ord}(g,x) = i$. Let f^{-1} be the multiplicative inverse of f in K. Then $\mathsf{ord}(f * f^{-1}, x) = 0$ and hence $\mathsf{ord}(f^{-1}, x) = -i$ and finally $\mathsf{ord}(g * f^{-1}, x) = 0$. Let $(f * f^{-1})(x) = \alpha$ and $(g * f^{-1})(x) = \beta$. By Property 3a, $\alpha, \beta \notin \{0, \infty\}$, and since x is a rational point, $\alpha, \beta \in \mathcal{F}_q$. Thus we find that $(\beta f * f^{-1} - \alpha g * f^{-1})(x) = 0$. Thus $\mathsf{ord}(\beta f * f^{-1} - \alpha g * f^{-1}, x) < 0$ and so $\mathsf{ord}(\beta f - \alpha g, x) < i$ as required. □

Lemma 16 *Given functions $\phi_1, \ldots, \phi_p$ of distinct orders at $x_0 \in \mathcal{X}$ satisfying $\phi_j \in L_{j+g-1, x_0}$ and a rational point $x_i \ne x_0$, there exist functions $\psi_1, \ldots, \psi_p \in K$ with $\mathsf{ord}(\psi_j, x_i) \le 1 - j$ and such that there exist $\alpha_{x_i, j_1, j_3} \in \mathcal{F}_q$ for $1 \le j_1, j_3 \le p$ such that $\phi_{j_1} = \sum_{j_3=1}^{p} \alpha_{x_i, j_1, j_3} \psi_{j_3}$.*

Proof: We prove a stronger statement by induction on p: If $\phi_1, \ldots, \phi_p$ are linearly independent (over $\mathcal{F}_q$) functions such that $\mathsf{ord}(\phi_j, x_i) \le m$ for $j \in [p]$, then there are functions $\psi_1, \ldots, \psi_p$ such that $\mathsf{ord}(\psi_j, x_i) \le m + 1 - j$ that generate the ϕ_j's over $\mathcal{F}_q$. Note that this will imply our lemma as $\phi_1, \phi_2, \ldots, \phi_p$ are linearly independent using Property 3(c) and the fact that the ϕ_j's have *distinct* pole orders at x_0. W.l.o.g. assume that ϕ_1 is a function with largest order at x_i. We let $\psi_1 = \phi_1$. Applying Lemma 15 to every pair (ϕ_1, ϕ_j) for $2 \le j \le p$, we obtain functions ϕ'_j such that $\phi'_j = \phi_j$ if $\mathsf{ord}(\phi_j, x_i) < \mathsf{ord}(\phi_1, x_i)$, or $\phi'_j = \alpha_j \phi_1 + \beta_j \phi_j$ for $\alpha_j, \beta_j \in \mathcal{F}_q - \{0\}$ if $\mathsf{ord}(\phi_j, x_i) = \mathsf{ord}(\phi_1, x_i)$. Hence for $2 \le j \le p$, ϕ_1, ϕ'_j generate ϕ_j. Now $\phi'_2, \phi'_3, \ldots, \phi'_p$ are linearly independent (since $\phi_1, \phi_2, \ldots, \phi_p$ are) and $\mathsf{ord}(\phi'_j, x_i) \le m - 1$ for $2 \le j \le p$, so the inductive hypothesis applied to the functions $\phi'_2, \ldots, \phi'_p$ now yields $\psi_2, \ldots, \psi_p$ as required. □

We are now ready to express condition (i) on (x_i, y_i) being a zero of order at least r. Using the above lemma and (1), we know that $Q(x,y)$ has the form

$$Q(x,y) = \sum_{j_2=0}^{s} \sum_{j_3=1}^{l-g+1} \sum_{j_1=1}^{l-g+1-j_2\alpha} q_{j_1,j_2} \alpha_{x_i,j_1,j_3}\, \psi_{j_3,x_i}(x) y^{j_2}.$$

The shifting to y_i is achieved by defining $Q^{(i)}(x,y) \stackrel{\rm def}{=} Q(x, y + y_i)$. The terms in $Q^{(i)}(x,y)$ that are divisible by y^p contribute p towards the multiplicity of $(x_i, 0)$ as a zero of $Q^{(i)}$, or, equivalently, the multiplicity of (x_i, y_i) as a zero of Q. We have

$$Q^{(i)}(x,y) = \sum_{j_4=0}^{s} \sum_{j_3=1}^{l-g+1} q^{(i)}_{j_3,j_4} \psi_{j_3,x_i}(x) y^{j_4}, \qquad (2)$$

where

$$q^{(i)}_{j_3,j_4} \stackrel{\rm def}{=} \sum_{j_2=j_4}^{s} \sum_{j_1=1}^{l-g+1-\alpha j_2} \binom{j_2}{j_4} y_i^{j_2-j_4} \cdot q_{j_1,j_2} \alpha_{x_i,j_1,j_3}.$$

Since $\mathsf{ord}(\psi_{j_3,x_i}, x_i) \le -(j_3 - 1)$, we can achieve our condition on (x_i, y_i) being a zero of multiplicity at least r by insisting that $q^{(i)}_{j_3,j_4} = 0$ for all $j_3 \ge 1$, $j_4 \ge 0$ such that $j_4 + j_3 - 1 < r$. Having developed the necessary machinery, we now proceed directly to the formal specification of our algorithm.

Implicit Parameters: n; $x_0, x_1, \ldots, x_n \in \mathcal{X}$; k; g.

Assumptions: We assume that we "know" functions $\{\phi_{j_1} \in K | j_1 \in [l - g + 1]\}$ of *distinct* orders at x_0 with $\mathsf{ord}(\phi_{j_1}, x_0) \le j_1 + g - 1$, as well as functions $\{\psi_{j_3,x_i} \in K | j_3 \in [l - g + 1], i \in [n]\}$ such that for any $i \in [n]$, the functions $\{\psi_{j_3,x_i}\}_{j_3}$ satisfy $\mathsf{ord}(\psi_{j_3,x_i}, x_i) \le 1 - j_3$. The notion of "knowledge" is explicit in the following two objects that we assume are available to our algorithm.

1. The set $\{\alpha_{x_i,j_1,j_3} \in \mathcal{F}_q | i \in [n], j_1, j_3 \in [l-g+1]\}$ such that for every i, j_1, $\phi_{j_1} = \sum_{j_3} \alpha_{x_i,j_1,j_3} \psi_{j_3,x_i}$. This assumption is a very reasonable one since Lemma 16 essentially describes an algorithm to compute this set given the ability to perform arithmetic in the function field K.
2. A polynomial-time algorithm to find roots (in K) of polynomials in $K[y]$ where the coefficients (elements of K) are specified as a formal sum of ϕ_{j_1}'s. (The cases for which such algorithms are known are described in [19, 18].)

The Algorithm:

- Inputs: $n, k, y_1, \ldots, y_n \in \mathcal{F}_q$.
- Algorithm Parameters: r, l:

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

and $l \stackrel{\text{def}}{=} rt-1$. (Recall that $\alpha \stackrel{\text{def}}{=} k+g-1$.)

Step 1: Find $Q[y] \in K[y]$ of the form $Q[y] = \sum_{j_2=0}^{s} \sum_{j_1=1}^{l-g+1-\alpha j_2} q_{j_1 j_2} \phi_{j_1} y^{j_2}$, i.e find values of the coefficients $\{q_{j_1,j_2}\}$ such that the following conditions hold:

1. At least one q_{j_1,j_2} is non-zero.
2. For every $i \in [n], \forall j_3, j_4, j_3 \geq 1, j_4 \geq 0$ such that $j_3 + j_4 \leq r$,

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

Step 2: Find all roots $h \in L_{k+g-1,x_0}$ of the polynomial $Q \in K[y]$. For each such h, check if $h(x_i) = y_i$ for at least t values of $i \in [n]$, and if so, include h in output list. (This step can be performed by either completely factoring Q using algorithms presented in [19], or more efficiently by using the root-finding algorithm of [18].)

4.3 Analysis of the Algorithm

We start by looking at $Q[h]$. Recall that for any $h \in K$, $Q[h] \in K$. By the condition (2) which we imposed on Q, we have $Q[h] \in L_{l,x_0}$ whenever $h \in L_{k+g-1,x_0}$.

Lemma 17 *For $i \in [n]$, if $h \in K$ satisfies $h(x_i) = y_i$, then* $\mathsf{ord}(Q[h], x_i) \leq -r$.

Proof: We have, for any such i, $Q[h](x) = Q(x, h(x)) = Q^{(i)}(x, h(x) - y_i) = Q^{(i)}(x, h(x) - h(x_i))$ and using (2), this yields

$$Q[h](x) = \sum_{j_4=0}^{s} \sum_{j_3=1}^{l-g+1} q^{(i)}_{j_3,j_4} \psi_{j_3,x_i}(x)(h(x) - h(x_i))^{j_4}.$$

Since $q^{(i)}_{j_3,j_4} = 0$ for $j_3 + j_4 \leq r$, $\mathsf{ord}(\psi_{j_3,x_i}, x_i) \leq 1 - j_3$, and if $h^{(i)} \in K$ is defined by $h^{(i)}(x) \stackrel{\text{def}}{=} h(x) - h(x_i)$, then $\mathsf{ord}((h^{(i)})^{j_4}, x_i) \leq -j_4$, we get $\mathsf{ord}(Q[h], x_i) \leq -r$ as desired. □

Lemma 18 *If $h \in L_{k+g-1,x_0}$ is such that $h(x_i) = y_i$ for at least t values of $i \in [n]$ and $rt > l$, then $y - h$ divides $Q[y] \in K[y]$.*

Proof: Using Lemma 17, we get $\sum_{i \in [n]} \mathsf{ord}(Q[h], x_i) \leq -rt < -l$. Since $Q[h] \in L_{l,x_0}$, we have $\sum_{x \in \overline{X}} \mathsf{ord}(Q[h], x) < 0$, implying $Q[h] \equiv 0$. Thus h is a root of $Q[y]$ and hence $y - h$ divides $Q[y]$. □

Lemma 19 *If $n\binom{r+1}{2} < \frac{(l-g)(l-g+2)}{2\alpha}$, then a $Q[y]$ as sought in Step 1 does exist (and can be found in polynomial time by solving a linear system).*

Proof: The proof follows that of Lemma 5. The computational task in Step 1 is once again that of solving a homogeneous linear system. A non-trivial solution exists as long as the number of unknowns exceeds the number of constraints. The number of constraints in the linear system is $n\binom{r+1}{2}$, while the number of unknowns equals

$$\sum_{j_2=0}^{s} (l - g + 1 - \alpha j_2) \geq \frac{(l-g)(l-g+2)}{2\alpha}. \qquad \square$$

Lemma 20 *If n, k, t, g satisfy $t^2 > (k+g-1)n$, then for the choice of r, l made in the algorithm, $n\binom{r+1}{2} < \frac{(l-g)(l-g+2)}{2(k+g-1)}$ and $rt > l$ both hold.*

Proof: Analogous to the proof of Lemma 6. □

Our main theorem now follows from Lemmas 18-20.

Theorem 21 *Let $\mathcal{C}$ be an $[n, k, d]_q$ algebraic-geometric code over an algebraic function field K of genus g (with $d = n - k - g + 1$). Then there exists a polynomial time list decoding algorithm for $\mathcal{C}$ that works for up to $e < n - \sqrt{n(k+g-1)} = n - \sqrt{n(n-d)}$ errors (provided the assumptions of the algorithm of Section 4.2 are satisfied).*

References

[1] S. Ar, R. Lipton, R. Rubinfeld and M. Sudan. Reconstructing algebraic functions from mixed data. *SIAM Journal on Computing*, 28(2):488-511, 1999.

[2] E. R. Berlekamp. *Algebraic Coding Theory*. McGraw Hill, New York, 1968.

[3] E. R. Berlekamp. Bounded Distance $+1$ Soft-Decision Reed-Solomon Decoding. *IEEE Transactions on Information Theory*, 42(3):704-720, 1996.

[4] R. E. BLAHUT. *Theory and Practice of Error Control Codes.* Addison-Wesley, Reading, Massachusetts, 1983.

[5] I. I. DUMER. Two algorithms for the decoding of linear codes. *Problems of Information Transmission*, 25(1):24-32, 1989.

[6] P. ELIAS. List decoding for noisy channels. Technical Report 335, Research Laboratory of Electronics, MIT, 1957.

[7] P. ELIAS. Error-correcting codes for list decoding. *IEEE Transactions on Information Theory*, 37:5-12. 1991.

[8] P. GEMMELL AND M. SUDAN. Highly resilient correctors for multivariate polynomials. *Information Processing Letters*, 43(4):169-174, 1992.

[9] O. GOLDREICH, R. RUBINFELD AND M. SUDAN. Learning polynomials with queries: The highly noisy case. *FOCS*, 1995.

[10] D. GRIGORIEV. Factorization of Polynomials over a Finite Field and the Solution of Systems of Algebraic Equations. Translated from Zapiski Nauchnykh Seminarov Lenningradskogo Otdeleniya Matematicheskogo Instituta im. V. A. Steklova AN SSSR, 137:20-79, 1984.

[11] J. JUSTESEN. Bounds on list decoding of MDS codes. Manuscript, April 1998.

[12] E. KALTOFEN. A Polynomial-Time Reduction from Bivariate to Univariate Integral Polynomial Factorization. *FOCS*, 1982.

[13] E. KALTOFEN. Polynomial factorization 1987–1991. *LATIN '92*, I. Simon (Ed.) Springer LNCS, v. 583:294-313, 1992.

[14] F. J. MACWILLIAMS AND N. J. A. SLOANE. *The Theory of Error-Correcting Codes.* North-Holland, Amsterdam, 1981.

[15] W. W. PETERSON. Encoding and error-correction procedures for Bose-Chaudhuri codes. IRE Transactions on Information Theory, IT-60:459-470, 1960.

[16] J. RADHAKRISHNAN. *Personal communication*, January, 1996.

[17] R. M. ROTH AND G. RUCKENSTEIN. Efficient decoding of Reed-Solomon codes beyond half the minimum distance. Manuscript, August 1998. Submitted to *IEEE Transactions on Information Theory.*

[18] M. A. SHOKROLLAHI. Computing roots of polynomials over function fields of curves. Manuscript, June 1998.

[19] M. A. SHOKROLLAHI AND H. WASSERMAN. Decoding algebraic-geometric codes beyond the error-correction bound. *STOC*, 1998.

[20] V. M. SIDELNIKOV. Decoding Reed-Solomon codes beyond $(d-1)/2$ and zeros of multivariate polynomials. *Problems of Information Transmission*, 30(1):44-59, 1994.

[21] H. STICHTENOTH. *Algebraic Function Fields and Codes.* Springer-Verlag, Berlin, 1993.

[22] M. SUDAN. Decoding of Reed-Solomon codes beyond the error-correction bound. *Journal of Complexity*, 13(1):180-193, 1997.

[23] M. SUDAN. Decoding of Reed-Solomon codes beyond the error-correction diameter. *Proceedings of the 35th Annual Allerton Conference on Communication, Control and Computing*, 1997.

[24] M. A. TSFASMAN, S. G. VLĂDUT AND T. ZINK. Modular curves, Shimura curves, and codes better than the Varshamov-Gilbert bound. *Math. Nachrichten*, 109:21-28, 1982.

[25] J. H. VAN LINT. *Introduction to Coding Theory.* Springer-Verlag, New York, 1982.

[26] A. VARDY. Algorithmic complexity in coding theory and the minimum distance problem. *STOC*, 1997.

[27] L. WELCH AND E. R. BERLEKAMP. Error correction of algebraic block codes. *US Patent* Number 4,633,470, issued December 1986.