Improved Decoding of Reed-Solomon and Algebraic-Geometry Codes

Venkatesan Guruswami* Madhu Sudan*

May 11, 1999

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 [27], 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 [21].

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-geometry 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-geometry codes extends the methods of [24] 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 may be of use in soft-decision decoding algorithms for Reed-Solomon codes.

Keywords: Error-correcting codes, Reed-Solomon codes, Algebraic-Geometry codes, Decoding algorithms, List decoding, Polynomial time algorithms.

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

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[1]. We focus on linear codes so that the set of codewords form a linear subspace of Σ^N. Reed-Solomon codes are a classical, and commonly used, construction of linear error-correcting codes that yield $[N = n, K = k+1, D = n-k]_q$ 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 at most 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 information 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 [21] manages to solve this problem in polynomial time, as long as $e < \frac{N-K+1}{2}$ (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 [2, 19] for a description) achieve such running time bounds. It is also easily seen that if $e \geq \frac{N-K+1}{2}$ 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 [7] (see also [8]) 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, for a family of $[N = n, K = k+1, D = n-k]_q$ Reed-Solomon codes, we can study $\epsilon = e/n$ as a function of $\kappa = (k+1)/n \approx k/n$. Till recently, no significant

[1] Usually an error correcting code is defined as a set of codewords, but for ease of exposition we describe it in terms of the underlying mapping, which also specifies the encoding method, rather than just the set of codewords.

benefits were achieved using the list decoding approach to recover from errors. The only improvements known over the algorithm of [21] were decoding algorithms due to Sidelnikov [25] and Dumer [6] which correct $\frac{n-k}{2} + \Theta(\log n)$ errors, i.e., achieve $\epsilon = (1-\kappa)/2 + o(1)$. Recently, Sudan [27], building upon previous work of Ar et al. [1], presented a polynomial time list decoding algorithm for Reed-Solomon codes correcting more than $(n-k)/2$ errors, provided $k < n/3$. The exact description of the number of errors ϵ_κ corrected by this algorithm is rather complicated and can be found in [28] or Figure 1. One lower bound on the number of errors corrected is $n - \sqrt{2kn}$, thus achieving $\epsilon = \epsilon_\kappa \geq 1 - \sqrt{2\kappa}$. A more efficient list decoding algorithm, running in time $O(n^2 \log^2 n)$, correcting the same number of errors has been given by Roth and Ruckenstein [23]. 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 [21]. This case is of interest since applications in practice tend to use codes of high rates.

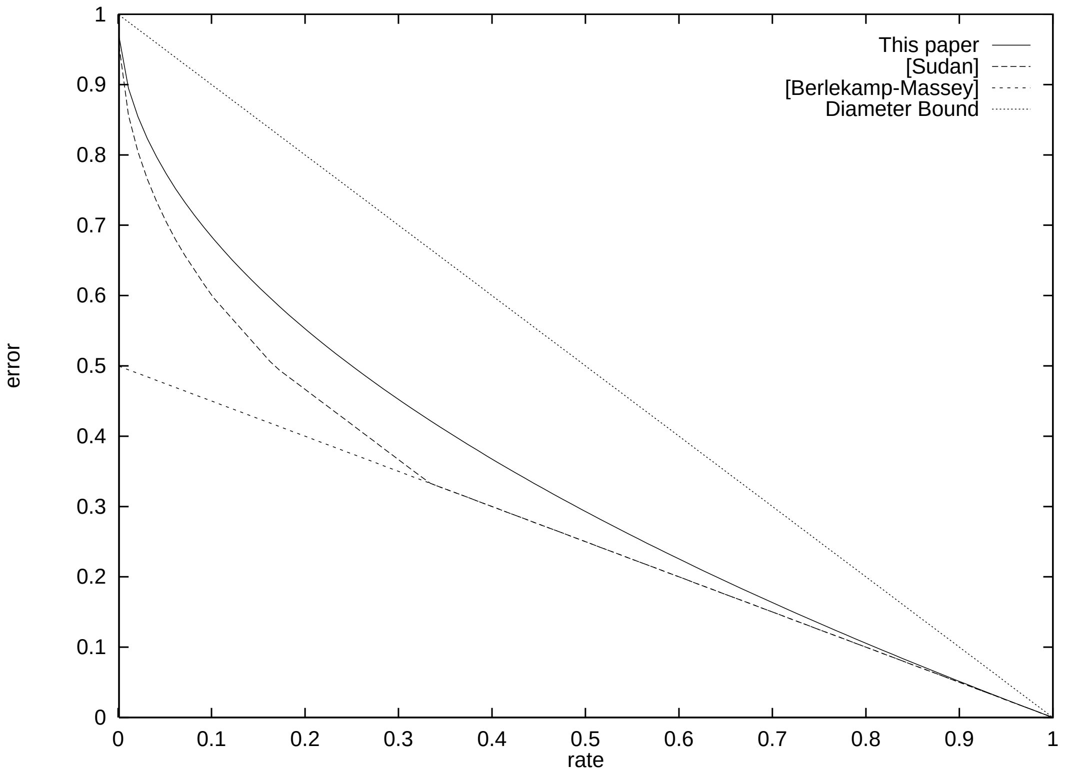

Figure 1: Error-correcting capacity plotted against the rate of the code for known algorithms.

In this paper we present a new polynomial-time algorithm for list-decoding of Reed-Solomon codes (in fact Generalized Reed-Solomon codes, to be defined in Section 2) that corrects up to (exactly) $\left\lceil n - \sqrt{nk} - 1 \right\rceil$

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 [21]. (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 [27] 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 [27] 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 [14, 17, 18]. 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. [13], shows that the number of solutions to the list decoding problem for a code with block length n and minimum distance d, is bounded by a polynomial in n as long as $e < n - \sqrt{n(n-d)}$. (A similar result has also been shown by Radhakrishnan [22].) 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 [16] has obtained upper bounds on the maximum number of errors $e = e_{c,d,n}$ for which the output of a list decoding algorithm can be guaranteed to have at most c solutions, for 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, and not optimizing the runtime of any of our algorithms.

Extensions to Algebraic-Geometry Codes Algebraic-geometry 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 [29] (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-geometry codes are typically based on decoding algorithms for Reed-Solomon codes. In particular, Shokrollahi and Wasserman [24] generalize the algorithm of Sudan [27] for the case of algebraic-geometry 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-geometry codes and obtain a list decoding algorithm correcting an $[n,k,d]_q$ algebraic-geometry 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[2]: 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 may be of interest in "soft-decision" decoding of Reed-Solomon codes.

2 Generalized 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 (Generalized Reed-Solomon codes) *For parameters n, k and a field F of cardinality q, a vector $\vec{\alpha}$ of* distinct *elements $\alpha_1, \alpha_2, \ldots, \alpha_n \in F$ (hence we need $n \leq q$), and a vector $\vec{v}$ of* non-zero *elements $v_1, \ldots, v_n \in F$, the Generalized Reed-Solomon code $\mathrm{GRS}_{F,n,k,\vec{\alpha},\vec{v}}$, is the function mapping the*

[2] The evolution of the solution to the "curve-fitting" problem is somewhat interesting. The initial solutions of Peterson [21] did not explicitly solve the curve fitting problem at all. The solution provided by Welch and Berlekamp [32, 3] do work in this setting, even though the expositions there do not mention the curve fitting problem (see in particular, the description in [12]). Their problem statement, however, disallows repeated values of x_i. Sudan's [27] 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)!

messages F^{k+1} to code space F^n, given by $\mathrm{GRS}_{F,n,k,\vec{\alpha},\vec{v}}(\vec{m})_j = v_j \cdot \sum_{i=0}^{k} \vec{m}_{i+1}(\alpha_j)^i$, for $\vec{m} \in F^{k+1}$ and $1 \leq j \leq n$.

Problem 1 (Generalized Reed-Solomon decoding)

INPUT: *Field F, n, k, $\vec{a}, \vec{v} \in F^n$ specifying the code $\mathrm{GRS}_{F,n,k,\vec{\alpha},\vec{v}}$. A vector $\vec{y} \in F^n$ and error parameter e.*

OUTPUT: *All messages $\vec{m} \in F^{k+1}$ such that $\Delta(\mathrm{GRS}_{F,n,k,\vec{\alpha},\vec{v}}(\vec{m}), \vec{y}) \leq e$.*

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 $i \in [n]$.*

The following proposition is easy to establish:

Proposition 2 *The generalized Reed-Solomon decoding problem reduces to the polynomial reconstruction problem.*

Proof: It is easily verified that the instance $(F, n, k, \vec{\alpha}, \vec{v}, \vec{y}, e)$ of the GRS decoding problem reduces to the instance $(k, n-e, n, \{(\alpha_i, y_i/v_i)\}_{i=1}^n)$ of the polynomial reconstruction problem. ▌

2.1 Informal description of the algorithm

Our algorithm is based on the algorithm of [27], 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 [27] 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 $Q_{\alpha,\beta}(x, y) = Q(x + \alpha, y + \beta)$. Observe that the following explicit relation between the coefficients $\{q_{ij}\}$ of Q and 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 3 (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 $(w_1, w_2)\text{-wt-deg}(Q)$, 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.

Algorithm *Poly-Reconstruct*:

Inputs: $n, k, t, \{(x_i, y_i)\}_{i=1}^{n}$, where $x_i, y_i \in F$.

Step 0: Compute parameters r, l such that

$$rt > l \text{ and } n\binom{r+1}{2} < \frac{l(l+2)}{2k}$$

In particular, set

$$\begin{aligned} r &\stackrel{\text{def}}{=} 1 + \left\lfloor \frac{kn + \sqrt{k^2n^2 + 4(t^2 - kn)}}{2(t^2 - kn)} \right\rfloor \\ l &\stackrel{\text{def}}{=} rt - 1 \end{aligned}$$

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.

End *Poly-Reconstruct*

2.3 Correctness of the Algorithm

We now prove the correctness of our algorithm. In Lemmas 4 and 5, Q can be any polynomial returned in Step 1 of the algorithm.

Lemma 4 *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 5 *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 4, 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 6 *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

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

$$
\begin{aligned}
&\geq \left(\left\lfloor \frac{l}{k} \right\rfloor + 1\right)\left(l + 1 - \frac{l}{2}\right) \\
&\geq \frac{l}{k} \cdot \frac{l+2}{2},
\end{aligned}
$$

and the result follows. ▮

Lemma 7 *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 8 *Algorithm* Poly-Reconstruct *on inputs n, k, t and the points $\{(x_i, y_i) : 1 \leq i \leq n\}$, correctly solves the polynomial reconstruction problem provided $t > \sqrt{kn}$.*

Proof: Follows from Lemmas 5, 6 and 7. ▮

We can also infer an upper bound on the number of codewords within radius $e < n - \sqrt{kn}$ in a Generalized Reed-Solomon code. This bound is already known even for general (even non-linear codes) [13, 22]. Our result can be viewed as a constructive proof of this bound for the specific case of Generalized Reed-Solomon codes.

Proposition 9 *The number of codewords that lie within an Hamming ball of radius $e < n - \sqrt{kn}$ in an $[n, k+1, d]_q$ Generalized Reed-Solomon code is $O(\sqrt{kn^3})$ (which is in turn $O(n^2)$).*

Proof: By Lemma 5, the number M of such codewords is at most the degree $\deg_y(Q)$ of the bivariate polynomial Q in y. Since the $(1,k)$-weighted degree of Q is at most l, $\deg_y(Q) \leq \lfloor l/k \rfloor$. Choosing $t = \lfloor \sqrt{kn} \rfloor + 1$ (which corresponds to the largest permissible value of the radius e), we have, by the choice of l, that

$$M = O(\frac{l}{k}) = O(\frac{knt}{k}) = O(\sqrt{kn^3}),$$

as desired. ▮

Corollary 10 *For a family of constant (relative) rate κ Generalized Reed-Solomon codes, the number of codewords in a Hamming ball of (relative) radius $\varepsilon = 1 - (1+\gamma)\sqrt{\kappa}$, for any constant $\gamma > 0$, is $O(1/\gamma^2)$.*

2.4 Runtime of the Algorithm

We now verify that the algorithm above can be implemented to run efficiently (in polynomial in n time) and also provide rough (but explicit) upper bounds on the number of operations it performs.

Proposition 11 *The algorithm above can be implemented to run using $O(\max\{\frac{k^3n^6t^6}{(t^2-kn)^6}, \frac{t^6}{k^3}\})$ field operations over F, provided $|F| \leq 2^n$.*[3]

Proof: (Sketch) The homogeneous system of equations solved in Step 1 of the algorithm clearly has at most $O(l^2/k)$ unknowns (since $\deg_y(Q) \leq \lfloor l/k \rfloor$ and $\deg_x(Q) \leq l$). Hence using standard methods, Step 1 can be implemented using $O((l^2/k)^3) = O(l^6/k^3)$ field operations. We claim that this is the dominant portion of the runtime and that Step 2 can be implemented to run within this time using standard bivariate polynomial factorization techniques. We sketch some details on the implementation of Step 2 below.

To implement Step 2, we first compute the *discriminant* $T(x) = \mathrm{disc}_y(Q(x,y))$ of $Q(x,y)$ with respect to y (treating it as a polynomial in y with coefficients in $F[X]$). Therefore $T \in F[X]$, and also $\deg(T) \leq 2d_xd_y$ where d_x, d_y are the degrees of Q in x and y respectively. This bound on the degree of T follows easily from the definition of the discriminant (see for instance [5]), and it is also easy to prove that the discriminant T can be computed in $O(d_xd_y^4)$ field operations.

Next we find an $\alpha \in F$ such that $T(\alpha) \neq 0$. This can be done deterministically by trying out an arbitrary set of $(2d_xd_y + 1)$ field elements because of the bound we know on the degree of T. Now, by the definition of the discriminant, for such an α, $Q(\alpha, y)$ is square-free as an element of $F[Y]$.

[3] In this analysis as well as the rest of the paper, we use the big-Oh notation to hide constants. We stress that these are universal constants and not functions of the field size $|F|$.

We then compute the shifted polynomial $Q'(x,y) \stackrel{\text{def}}{=} Q(x+\alpha, y)$, so that $Q'(0,y)$ is square-free. Now we use the algorithm in [11] that can compute all roots $p \in F[x]$ of a bivariate polynomial $R(x,y)$ such that $R(0,y) \in F[Y]$ is square-free, in $O(k^2 \mathrm{deg_y}^2(R))$ time. This gives us a list of all polynomials $p'(x)$ such that $y - p'(x)$ divides $Q'(x,y)$; by computing $p(x) = p'(x-\alpha)$ for each such p' gives us the desired list of roots $p(x)$ of $Q(x,y)$. It is clear that once α is computed, all the above steps can be performed in at most $O(k^2 d_y^2)$ field operations.

Summing up, Step 2 can be performed using

$$O(d_x d_y^4 + d_x d_y + k^2 d_y^2) = O\Big(\frac{l^5}{k^4} + l^2\Big) = O\Big(\frac{l^5}{k^3}\Big)$$

field operations.

The entire algorithm can thus be implemented to run in $O(l^6/k^3)$ field operations and since

$$l = O(\max\{\frac{knt}{t^2 - kn}, t\})$$

the claimed bound on the runtime follows. ▮

Theorem 12 *The polynomial reconstruction problem can be solved in time $O(n^{15})$, provided $t > \sqrt{kn}$, for any field F of cardinality at most 2^n. Furthermore, if $t^2 = (1+\delta)kn$, then the problem can be solved in time $O(n^3 \delta^{-6})$.*

Proof: Follows from Proposition 11 and Theorem 8. ▮

Corollary 13 *Given a family of Generalized Reed-Solomon codes of constant message rate κ, an error-rate of $\epsilon = 1 - \sqrt{\kappa}$ can be list-decoded in time $O(n^{15})$. When $\epsilon < 1 - \sqrt{\kappa}$, then the decoding time reduces to $O(n^3(1 - \epsilon - \sqrt{\kappa})^{-12}) = O(n^3)$.*

3 Some Consequences

First of all, since the classical Reed-Solomon codes are simply a special case of Generalized Reed-Solomon codes, Corollary 13 above holds for Reed-Solomon codes as well. We now describe some other 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 family of codes called alternant codes that includes a wide family of codes such as BCH codes, Goppa codes etc.

Definition 14 (Alternant Codes ([19], §12.2)) *For positive integers m, k_0, n, prime power q, the field $F = \mathrm{GF}(q^m)$, a vector $\vec{\alpha}$ of distinct elements $\alpha_1, \ldots, \alpha_n \in \mathrm{GF}(q^m)$, and a vector $\vec{v}$ of nonzero elements $v_1, \ldots, v_n \in \mathrm{GF}(q^m)$, the alternant code $\mathcal{A}_{q,n,k_0,\vec{\alpha},\vec{v}}$ comprises of all the codewords of the Generalized Reed-Solomon code defined by $\mathrm{GRS}_{F,n,k_0,\vec{\alpha},\vec{v}}$ that lie in $GF(q)^n$.*

Since the Generalized Reed-Solomon code has distance exactly $n - k_0 + 1$, it follows that the respective alternant code, being a subcode of the Generalized Reed-Solomon code, has distance at least $n - k_0 + 1$. We term this the *designed distance* $d' = n - k_0 + 1$ 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 15 *Let $\mathcal{A}$ be an $[n, k+1, d]_q$ alternant code with designed distance d' (and thus satisfying $\frac{d}{m} \leq d' \leq d$). 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 for alternant codes given in classical texts seem to correct $d'/2$ errors. For the more restricted BCH codes, there are algorithms that decode beyond half the designed distance (cf. [9] and also [4, Chapter 9]).

3.2 Errors and Erasures decoding

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 $n' \leq n$ 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 12 yields that e errors can be corrected provided $e < n' - \sqrt{n'k}$. Thus we get:

Theorem 16 *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*. Applying Theorem 12 (in particular by applying the theorem on point sets where the x_i's are not distinct) we get the following result.

Theorem 17 *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 [31] 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 have any 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 $w_i\rho$ at the point (x_i, y_i). Thus we would now have $\sum_{i=1}^{n} \binom{\rho w_i+1}{2}$ 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 $\sum_{i\in S} \rho w_i$ is greater than $(1, k)$-wt-deg(Q). Optimizing over the weighted degree of Q yields the following theorem.

Theorem 18 *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-Geometry Codes

We now describe the extension of our algorithm to the case of algebraic-geometry codes. Our extension follows along the lines of the algorithm of Shokrollahi and Wasserman [24]. Our extension shows that the algebra of the previous section extends to the case of algebraic function fields, yielding an approach to the list decoding problem for algebraic-geometry codes. In particular it reduces the decoding problem to some basis computations in an algebraic function field and to a factorization (actually root-finding) problem over the algebraic function field. However neither of these tasks is known to be solvable efficiently given only the generator matrix of the linear code. It is conceivable however that given some polynomial amount of additional information about the linear code, one can solve both parts efficiently. In fact for the former task we show that this is indeed the case; for the latter part we are not aware of any such results. For certain representations of some function fields, Shokrollahi and Wasserman [24] give factorization algorithms that run in time polynomial in the representation of the field. It is not however still clear if these representations are of size that is bounded by some polynomial in the block length of the code. Thus the results of this section are best viewed as reductions of the list-decoding problem to a factorization problem over algebraic function fields.

Much of the work of this section is in ferreting out the axioms satisfied by these constructions, so as to justify our steps. We do so in Section 4.1. Then we present our algorithm for list decoding modulo some algorithmic assumptions about the underlying structures. Under these assumptions, 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 over the result of [24], which corrects up to $e < n - \sqrt{2n(n-d)} - g + 1$ errors.

4.1 Definitions

An algebraic-geometry code is built over a structure termed an algebraic function field. Definitions and basic properties of these codes can be found in [15, 26]; for purposes of self-containment and ease of exposition, we now develop a slightly different notation to express our results.

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 $\emptyset_{i,x} = \{f | \mathsf{ord}(f,x) \leq i\}$ 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 $L_{i,x} = \emptyset_{i,x} \cap (\cap_{y \in \overline{\mathcal{X}} - \{x\}} \emptyset_{0,y})$, 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 19 *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 $q \geq 49$ [29].

4.2 The Decoding Algorithm

We now describe the extension of our algorithm to the case of algebraic-geometry codes. As usual we will try to describe the data points $\{(x_i, y_i)\}$ by some polynomial Q. We follow [24] 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 [24] 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 [24], 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 $\alpha \stackrel{\text{def}}{=} k + g - 1$ 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 $1 \leq i \leq n$, and (ii) $\mathsf{ord}(Q[f], x_0) \leq l$ 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 $\mathsf{ord}(\phi_j, x_0) \leq j + g - 1$ and such that $\mathsf{ord}(\phi_j, x_0) < \mathsf{ord}(\phi_{j+1}, x_0)$. Let $s \stackrel{\text{def}}{=} \left\lfloor \frac{l-g}{\alpha} \right\rfloor$. 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 20 *For every $f, g \in K$ and $x \in \mathcal{X}$ with $\mathsf{ord}(f, x) = \mathsf{ord}(g, x)$, there exist $\alpha_0, \beta_0 \in \mathcal{F}_q \setminus \{0\}$, such that*

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

Proof: Let $\mathsf{ord}(f, x) = \mathsf{ord}(g, x) = i$ and 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 21 *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 $x_i \neq x_0$, 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, by assumption $\mathsf{ord}(\phi_1, x_i) \le m$. We let $\psi_1 = \phi_1$. Now, for $2 \le j \le p$, set $\phi'_j = \phi_j$ if $\mathsf{ord}(\phi_j, x_i) < \mathsf{ord}(\phi_1, x_i)$. If $\mathsf{ord}(\phi_j, x_i) = \mathsf{ord}(\phi_1, x_i)$, using Lemma 20 to the pair (ϕ_1, ϕ_j) of functions, we get $\alpha_j, \beta_j \in \mathcal{F}_q - \{0\}$ such that the function $\phi'_j = \alpha_j \phi_1 + \beta_j \phi_j$ satisfies $\mathsf{ord}(\phi'_j, x_i) < \mathsf{ord}(\phi_1, x_i) \le m$. Since in this case $\phi_j = \beta_j^{-1} \phi'_j - \alpha_j \beta_j^{-1} \phi_1$, we conclude that in any case, for $2 \le j \le p$, $\psi_1 = \phi_1$ and 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 $Q^{(i)}(x, y) \stackrel{\text{def}}{=} Q(x, y + y_i)$. 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

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

where

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

Since $\mathsf{ord}(\psi_{j_3,x_i}, x_i) \leq -(j_3 - 1)$, 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 $j_3 \geq 1$, $j_4 \geq 0$ 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 $\mathsf{ord}(\phi_{j_1}, x_0) \leq j_1 + g - 1$, 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 $\mathsf{ord}(\psi_{j_3,x_i}, x_i) \leq 1 - j_3$. 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 21 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 [24, 11].)

The Algorithm:

Inputs: $n, k, y_1, \ldots, y_n \in \mathcal{F}_q$.

Step 0: Computer parameters r, l such that

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

(Recall that $\alpha \stackrel{\mathrm{def}}{=} k + g - 1$.) In particular set

$$\begin{aligned} r &\stackrel{\mathrm{def}}{=} 1 + \left\lfloor \frac{2gt + \alpha n + \sqrt{(2gt+\alpha n)^2 - 4(g^2-1)(t^2-\alpha n)}}{2(t^2 - \alpha n)} \right\rfloor, \\ l &\stackrel{\mathrm{def}}{=} rt - 1 \end{aligned}$$

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 [24], or more efficiently by using the root-finding algorithm of [11].)

The following proposition says that the above algorithm can be implemented efficiently modulo some (reasonable) assumptions.

Proposition 22 *Given the ability to perform field operations in the subset L_{l,x_0} of the function field K when elements are expressed as a formal combination of the ϕ_{j_1}'s for $j_1 \in [l-g+1]$, the above algorithm reduces the decoding problem of an $[n,k,d]_q$ algebraic geometry code (with designed distance $d' = n-k-g+1$) in time (measured in operations over K) at most $O(l^6/(n-d')^3 + nl^2)$ to a root-finding problem over the function field K of a univariate polynomial of degree at most $l/(n-d')$ with coefficients having pole order at most l, where $l = O(\max\{\frac{gt+n(n-d')}{t^2-n(n-d')}, t\})$.*

Proof: First of all, note that the computation of all the α_{x_i,j_1,j_3}'s can be done in $O(nl^2)$ operations over K. The system of equations set up in Step 1 has at most $l(l-g)/\alpha = O(l^2/(n-d'))$ unknowns, and hence can be solved in $O(l^6/(n-d')^3)$ operations (over $\mathcal{F}_q$). Also, it is clear that the degree of $Q \in K[Y]$ thus found is at most $(l-g)/\alpha = O(l/(n-d'))$ and that all coefficients of Q have at most l poles at x_0 and no poles elsewhere. The claimed result now follows once we note that

$$l = O\Big(\max\{\frac{gt+n(n-d')}{t^2-n(n-d')}, t\}\Big)$$

▮

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 (ii) which we imposed on Q, we have $Q[h] \in L_{l,x_0}$ whenever $h \in L_{k+g-1,x_0}$.

Lemma 23 *For $i \in [n]$, if $h \in K$ satisfies $h(x_i) = y_i$, then $\mathsf{ord}(Q[h], x_i) \le -r$.*

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

Lemma 24 *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 23, we get $\sum_{i\in[n]} \mathsf{ord}(Q[h], x_i) \le -rt < -l$. Since $Q[h] \in L_{l,x_0}$, we have $\sum_{x\in\overline{\mathcal{X}}} \mathsf{ord}(Q[h], x) < 0$, implying $Q[h] \equiv 0$. Thus h is a root of $Q[y]$ and hence $y - h$ divides $Q[y]$. ▮

Lemma 25 *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 6. 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

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

▮

Lemma 26 *If n, k, t, g satisfy $t^2 > (k + g - 1)n$, then for the choice of r, l made in the algorithm, $\frac{(l-g)(l-g+2)}{2\alpha} > n\binom{r+1}{2}$ and $rt > l$ both hold.*

Proof: The proof parallels that of Lemma 7. The condition $rt > l$ certainly holds since we pick $l \stackrel{\text{def}}{=} rt - 1$. Using $l = rt - 1$, the other constraint becomes

$$\frac{(rt-g)^2 - 1}{2\alpha} > n\binom{r+1}{2}$$

which simplifies to

$$r^2(t^2 - \alpha n) - (2gt + \alpha n)r + (g^2 - 1) > 0.$$

If $t^2 - \alpha n > 0$, it suffices to pick r to be an integer greater than the larger root of the above quadratic, and therefore picking

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

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

Our main theorem now follows from Lemmas 24-26 and the runtime bound proved in Proposition 22

Theorem 27 *Let $\mathcal{C}$ be an $[n,k,d]_q$ algebraic-geometry 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).*

5 Concluding Remarks

We have given a polynomial time algorithm to decode up to $1 - \sqrt{\kappa}$ errors for a rate κ Reed-Solomon code and generalized the algorithm for the broader class of Algebraic-Geometry codes. Our algorithm is able to correct a number of errors exceeding half the minimum distance for any rate.

A natural question not addressed in our work is more efficient implementation of the decoding algorithms. Extensions of the works of [23, 11] seem to be promising directions in this regard. An important step, that of solving the associated linear equations efficiently, has already been taken by [20]. However some important problems, such as efficient factorization algorithms for polynomials over function fields, remain unsolved.

The list decoding problem remains an interesting question and it is not clear what the true limit is on the number of efficiently correctable errors. Deriving better upper or lower on the number of correctable errors remains a challenging and interesting pursuit.

Acknowledgments

We would like to the anonymous referees for numerous comments which improved and clarified the presentation a lot. We would like to express our thanks to Elwyn Berlekamp, Peter Elias, Jorn Justesen, Ronny Roth, Amin Shokrollahi and Alex Vardy for useful comments on the paper.

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] H. COHEN. *A Course in Computational Algebraic Number Theory.* GTM 138, Springer Verlag, Berlin, 1993.

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

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

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

[9] G. L. FENG AND K. K . TZENG. A generalization of the Berlekamp-Massey algorithm for multisequence shift register synthesis with application to decoding cyclic codes. *IEEE Trans. Inform. Theory*, 37 (1991), pp. 1274-1287.

[10] G. D. FORNEY. Generalized Minimum Distance Decoding. *IEEE Trans. Inform. Theory*, Vol. 12, pp. 125-131, 1966.

[11] S. GAO AND M. A. SHOKROLLAHI. Computing roots of polynomials over function fields of curves. Manuscript, August 1998.

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

[13] O. GOLDREICH, R. RUBINFELD AND M. SUDAN. Learning polynomials with queries: The highly noisy case. *Proceedings of the 36th Annual IEEE Symposium on Foundations of Computer Science*, pages 294–303, 1995.

[14] 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.

[15] T. HϕHOLDT, J. H. VAN LINT AND R. PELLIKAAN. *Algebraic Geometry Codes*. In Handbook of Coding Theory, (V.S. Pless, W.C. Huffamn and R.A. Brualdi Eds.), Elsevier.

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

[17] E. KALTOFEN. A Polynomial-Time Reduction from Bivariate to Univariate Integral Polynomial Factorization. *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing*, pages 261-266, San Francisco, California, 5-7 May 1982.

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

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

[20] V. OLSHEVSKY AND M. A. SHOKROLLAHI. A displacement structure approach to efficient list decoding of algebraic geometric codes. *Proceedings of the 31st Annual ACM Symposium on Theory of Computing*, 1999, to appear.

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

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

[23] 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.*

[24] M. A. SHOKROLLAHI AND H. WASSERMAN. Decoding algebraic-geometric codes beyond the error-correction bound. *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, pages 241–248, 1998.

[25] 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.

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

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

[28] 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.

[29] 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.

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

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

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