Decoding of Reed Solomon codes beyond the error-correction bound

Madhu Sudan *

Abstract

We present a randomized algorithm which takes as input n distinct points $\{(x_i, y_i)\}_{i=1}^n$ from $F \times F$ (where F is a field) and integer parameters t and d and returns a list of all univariate polynomials f over F in the variable x of degree at most d which agree with the given set of points in at least t places (i.e., $y_i = f(x_i)$ for at least t values of i), provided $t = \Omega(\sqrt{nd})$. The running time is bounded by a polynomial in n. This immediately provides a maximum likelihood decoding algorithm for Reed Solomon Codes, which works in a setting with a larger number of errors than any previously known algorithm. To the best of our knowledge, this is the first efficient (i.e., polynomial time bounded) algorithm which provides error recovery capability beyond the error-correction bound of a code for any efficient (i.e., constant or even polynomial rate) code.

1 Introduction

1.1 Problem Statement

We consider the following problem.

*IBM Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, NY 10598, USA. email: `madhu@watson.ibm.com`.

Problem 1
Input: A field F; n distinct pairs of elements $\{(x_i, y_i)\}_{i=1}^{n}$ from $F \times F$; and integers d and t.

Output: A list of all functions $f : F \to F$ satisfying

$$f(x) \text{ is a polynomial in } x \text{ of degree at most } d \text{ with } |\{i|f(x_i) = y_i\}| \geq t \quad (1)$$

The above problem, and close variants, have been considered before in the context of coding theory and learning. The best threshold on t for which a polynomial time bounded algorithm to solve this problem was previously known is $t \geq \frac{n+d+1}{2}$. Notice that the ratio $t/n \to \frac{1}{2}$ if we fix d and let $n \to \infty$. In fact, when t satisfies this property then there is a unique function f satisfying (1). A particularly simple algorithm for this case is given by Berlekamp and Welch [4] (see, for instance, Gemmell and Sudan [9]).

In this paper we present an algorithm which solves the problem given above for $t = \Omega(\sqrt{nd})$. Notice that for fixed d, as $n \to \infty$, the fraction of agreement required by our algorithm approaches 0 (and not $\frac{1}{2}$ which is what previous algorithms seem to get). The algorithm is based on an algorithm of Ar et al. [1] for a restricted version of this problem.

The task of reconstructing polynomials describing a given set of points is relevant in the context of coding theory and we describe this context next. This task may also be of some relevance to computational complexity theory. We touch on this motivation very briefly in the conclusions.

1.2 Error-correcting Codes

For integers n, k, δ and a collection of symbols Σ, an $[n, k, \delta]$-code over Σ is a collection $\mathcal{C} \subset \Sigma^n$ of n-letter words over the alphabet Σ with $|\mathcal{C}| = |\Sigma|^k$ and the property that any two strings in $\mathcal{C}$ differ in at least δ places (i.e., the strings have Hamming Distance d). Given a code $\mathcal{C}$, the largest δ for which $\mathcal{C}$ is a $[n, k, \delta]$-code is referred to as the distance of the code. If τ satisfies $2\tau + 1 \leq \delta$, then an $[n, k, \delta]$-code $\mathcal{C}$ is also a τ-error-correcting code. The largest such value τ will be referred to as the error-correction bound of a code.

This terminology reflects that fact that given any string $s \in \Sigma^n$, there is at most one string $c \in \mathcal{C}$ which is within a Hamming distance of τ from s.

The codes of relevance to this paper are the Reed Solomon Codes. For a finite field F of size n and parameter d, the Reed Solomon Code is an $[n, d+1, n-d]$-code over F, whose codewords are the strings $\{(p(0), p(w), p(w^2), \ldots, p(w^{|F|-1}))\}_p$, where w is some fixed primitive element of F and p ranges over all polynomials of degree at most d over F (see, for instance, [18], page 86).

The algorithmic tasks of relevance to this paper are the tasks of "error-correction" and "maximum-likelihood decoding". The problem of τ-error-correction for an $[n, k, \delta]$-code is defined for $2\tau + 1 \leq \delta$ as follows: "Given a string $s \in \Sigma^n$, find a string $c \in \mathcal{C}$ which is within Hamming distance τ of s, if one such exists." Since $\mathcal{C}$ is a τ error-correcting code, the answer, if it exists, is unique. Notice that the problem is not defined for values of τ and δ (i.e., when $2\tau + 1 > d$) which may allow for non-unique answers. The maximum-likelihood decoding problem (see, for instance, [17] page 8) is set in a model where *a larger number of errors is considered less likely*, and is defined as follows: "Given a string $s \in \Sigma^n$, find a (the) codeword $c \in \mathcal{C}$ which is nearest to s (i.e., least Hamming distance away from s)." This problem is also sometimes referred to as the nearest codeword problem.

Previous work has focused mainly on the task of error-correction and algorithms are known for error-correction of many interesting codes. In particular, for Reed Solomon Codes, the τ-error-correction problem for a τ-error correcting code can be solved in polynomial time. In particular, the earlier mentioned solution of Berlekamp and Welch [4] works in this setting.

The case of recovering from error larger than the error-correction capacity of a code has not attracted the same amount of attention and significantly less is known about this problem. Since in this case the solution to the maximum-likelihood decoding problem may not be unique, it is not clear which solution is to be reported when the answer is not unique. Further, it is not clear why any algorithm would be able to (or should be allowed to) prefer any one solution over any other equally respectable solution.

However, it is possible to define a closely related problem which does not offer the algorithm any choice in its solutions. This problem, sometimes called the τ-reconstruction problem, is defined as follows: "Given a string

s, find *all* codewords $c \in \mathcal{C}$ that are within Hamming distance τ of s." This problem is also known as the list decoding problem. The reconstruction problem offers the solution to the maximum-likelihood decoding problem for a much larger range of τ than allowed by the τ-error correction problem. This is the problem tackled in this paper for the case of Reed Solomon Codes.

The τ-reconstruction problem is not a universal panacea for the maximum-likelihood problem. In fact by making the task enumerative (rather than picking one element from a large set, we want the whole set), the complexity requirements of the task go up. In particular the running time is lower bounded by the output size. Bounds on the output size of the reconstruction problem have been studied in the context of coding theory and a well known bound, due to S. Johnson (see [17], page 525), bounds the number of solutions by $\frac{n(n-\delta)}{2\tau^2-(2n)\tau+n(n-d)}$ for binary codes (i.e., codes over the alphabet $\{0,1\}$), provided the denominator is positive. For general codes, a simple bound can be shown by an inclusion-exclusion argument (see, for instance, [11]) which yields that the number of solutions to the τ-reconstruction problem is at most $2/\epsilon$, if $\tau = (1-\epsilon)\delta$, provided $\tau < n - \sqrt{2n(n-\delta)}$. (Another such bound is also known due to Goldreich, Rubinfeld and Sudan [11]. We do not describe this here.) However the inclusion-exclusion bound is not constructive, i.e., it does not provide a list of the $2/\epsilon$ codewords which may be the solution to the reconstruction problem.

It is reasonable to ask for a solution to the reconstruction problem which runs in polynomial time, when the output size in bounded. Here we solve the τ-reconstruction problem for Reed Solomon Codes for exactly the same values of the parameters δ and τ for which the inclusion-exclusion bound seems to work. Finding bounds which work for more general settings of δ and τ and finding solutions to the reconstruction problem which work in such settings remain open questions.

1.3 Previous work

As mentioned in the previous section, the τ-error correcting problem is well-studied and we will not describe past work in that problem here. The definition of the τ-reconstruction problem used here is based on definitions of Ar, Lipton, Rubinfeld and Sudan [1] and Goldreich, Rubinfeld and Sudan [11]. To the best of our knowledge, there are only two instances where the τ-reconstruction problem has found interesting solutions for some error-correcting code. The first, due to Goldreich and Levin [10], provides a solution for certain families of Hadamard Codes. Kushilevitz and Mansour [15], provide a variant of this algorithm which applies to the same codes. The second instance involves a generalization of the codes and algorithm given by [10], and is due to [11]. Both the codes given here are extremely low-rate codes. In fact, the rate (i.e., the ratio k/n), for these codes is $O(\frac{\log^{O(1)} n}{n})$ and thus a brute-force algorithm (running in time $|\Sigma|^{O(k)}$) is not too inefficient for these problems. The feature that makes the solutions of [10, 15, 11] interesting and efficient is that they work in time polynomial in k, using random access to an oracle describing the input s. This makes the solution interesting in some learning theoretic and cryptographic settings, but is however not very useful for coding theoretic applications (due to the low information rate). The techniques of Goldreich and Levin (which are inherited by [15, 11]) are interesting in that they manage to convert, modularly, the non-constructive bounds on the number of outputs (discussed earlier) into constructive ones. But their technique does not appear to generalize to the setting of Reed Solomon Codes (or any other high-rate codes).

Ar et al. [1] do provide some solutions to the reconstruction problem, but not in its full generality. In particular they restrict the nature of the input word s. For this restricted case they provide a solution to the reconstruction problem based on algebraic techniques (and in particular uses polynomial time solutions to the bivariate factoring problem). Our solution is a minor modification of their algorithm and analysis which manages to get around their restriction.

2 Algorithm

We now present our algorithm for solving the problem given in Section 1.1.

Definition 1 (weighted degree) *For weights $w_x, w_y \in \mathcal{Z}^+$, the (w_x, w_y)-weighted degree of a monomial $q_{ij}x^iy^j$ is $iw_x + jw_y$. The (w_x, w_y)-weighted degree of a polynomial $Q(x,y) = \sum_{ij} q_{ij}x^iy^j$ is the maximum, over the mono-*

mials with non-zero coefficients, of the (w_x, w_y)-weighted degree of the monomial.

Algorithm: /* Inputs: n, d, t; $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ */

A. /* Parameters l, m to be set later. */

B. Find *any* function $Q : F^2 \to F$ satisfying

$$\left.\begin{array}{l} Q(x,y) \text{ has } (1,d)\text{-weighted degree at most } m + ld, \\ \forall i \in [n],\ Q(x_i, y_i) = 0, \\ Q \text{ is not identically zero.} \end{array}\right\} \qquad (2)$$

C. Factor the polynomial Q into irreducible factors.

D. Output all the polynomials f such that $(y - f(x))$ is a factor of Q and $f(x_i) = y_i$ for at least t values of i from $[n]$.

Note: Step C above can be solved in randomized polynomial time with zero-sided error. If F is of characteristic zero, or if the running time is allowed to be polynomial in $|F|$, then the solution can be obtained deterministically. (See, for instance, [13].)

3 Analysis

In order to prove that the algorithm above can be run in polynomial time and works correctly, we need to prove the following set of claims. In all the following claims, we fix the set of pairs $\{(x_1, y_1), \ldots, (x_n, y_n)\}$.

Claim 2 *If a function $Q : F^2 \to F$ satisfying (2) exists, then one can be found in time* $\text{poly}(n)$.

Proof: Let $Q(x,y) = \sum_{j=0}^{l} \sum_{k=0}^{m+(l-j)d} q_{kj} x^k y^j$. Then we wish to find coefficients q_{jk} satisfying the constraints $\sum_{j=0}^{l} \sum_{k=0}^{m+(l-j)d} q_{kj} x_i^k y_i^j = 0$, for every $i \in [n]$. This is a linear system in the unknowns $\{q_{kj}\}$ and hence if a solution exists then one can be found in polynomial time. ▮

Claim 3 *If $(m+1)(l+1) + d\binom{l+1}{2} > n$ then there exists a function $Q(x,y)$ satisfying (2).*

Proof: First we observe that the linear system is homogenous. Therefore the setting $q_{kj} = 0$, satisfies all the linear constraints. However this does not satisfy (2), since Q would be identically zero. In order to show that a non-zero solution exists, we observe that the number of unknowns in the linear system that we wish to solve is $(m+1)(l+1) + d\binom{l+1}{2}$. Since this is more than n, we have a homogenous linear system with more variables than constraints and hence a non-zero solution exists. ▮

The following lemma is a special case of a general class of theorems known as Bezout's Theorem. Since we will be interested in tight behavior of the theorem, we use a version due to [1]. The proof is also from [1].

Claim 4 *If $Q(x,y)$ is a function satisfying (2) and $f(x)$ is a function satisfying (1) and $t > m + ld$, then $(y - f(x))$ divides $Q(x,y)$.*

Proof: Consider the function $p(x) \stackrel{\text{def}}{=} Q(x, f(x))$. This is a polynomial in x and we argue that it has degree at most $m + ld$. Consider any monomial $q_{kj} x^k y^y$ of $Q(x,y)$. Since Q has $(1,d)$-weighted degree at most $m + ld$, we have that $k + jd \le m + ld$. Thus the term $q_{kj} x^k (f(x)^j$ is a polynomial in x of degree at most $k + jd \le m + ld$. Thus $p(x)$ has degree at most $m + ld$ (since it is a sum polynomials of degree at most $m + ld$).

Now we argue that $p(x)$ is identically zero. Since $Q(x_i, f(x_i))$ is zero whenever $y_i = f(x_i)$, we have that $p(x_i)$ is zero for strictly greater than $m + ld$ points. Thus p has more zeroes than its degree and hence is identically zero, implying $Q(x, f(x)) \equiv 0$.

Now consider $Q(x,y)$ as a polynomial $Q_x(y)$ in y with coefficients from $F[x]$, the ring of polynomials in x. By the polynomial remainder theorem, we have that if $Q_x(\zeta) = 0$, then $(y-\zeta)$ divides $Q_x(y)$. Substituting $\zeta = f(x)$, yields the lemma. ∎

We are now ready to optimize the parameters m and l. For now we will ignore the fact that l and m have to be integers and fix this aspect later. Notice that we want

$$m + ld < t \text{ and } (m+1)(l+1) + d\binom{l+1}{2} > n.$$

Thus given a value of l, we can compute the smallest m for which the second condition holds, and that is $\frac{n+1-d\binom{l+1}{2}}{l+1} - 1$. Thus we find that t must be at least

$$\frac{n+1-d\binom{l+1}{2}}{l+1} - 1 + ld + 1 = \frac{n+1}{l+1} + \frac{dl}{2}.$$

We can now minimize the expression above as a function of the unknown parameter l to obtain the smallest value of t for which this algorithm will work. The minimum occurs when

$$\frac{-(n+1)}{(l+1)^2} + \frac{d}{2} = 0 \Rightarrow l = \sqrt{\frac{2(n+1)}{d}} - 1.$$

This setting yields

$$m \geq \sqrt{(n+1)d/2} - \sqrt{(n+1)d/2} + d/2 - 1 = d/2 - 1$$

$$\text{and } t > m + ld \geq d/2 - 1 + \sqrt{2(n+1)d} - d = \sqrt{2(n+1)d} - d/2 - 1.$$

Due to integrality issues we will lose a little bit in the following theorem.

Theorem 5 *Given a sequence of n distinct pairs $\{(x_i, y_i)\}_{i=1}^n$, where the x_is and y_is are elements of a field F, and integer parameters t and d, such that $t \geq d\lceil\sqrt{2(n+1)/d}\rceil - \lfloor d/2 \rfloor$, there exists an algorithm, which runs in time polynomial in n, that can find all the polynomials $f : F \rightarrow F$ of degree at most d, such that the number of point (x_i, y_i) such that $y_i = f(x_i)$ is at least t.*

Proof: We use the algorithm of Section 2 with $m = \lceil d/2 \rceil - 1$ and $l = \lceil \sqrt{(2(n+1)/d)} \rceil - 1$. It may be verified that this setting satisfies the condition $(m+1)(l+1) + d\binom{l+1}{2} \geq n+1$. Hence by Claim 3, Step 2 will return a function $Q(x, y)$ satisfying property (2). Furthermore, the setting of m and l also satisfies the condition $t > m + ld$. Thus Claim 4 guarantees that if f is a function satisfying (1), then $(y - f(x))$ will divide the polynomial Q returned in Step 2 and hence be one of our outputs. ▮

4 Bound on the number of polynomials

Here we give a new proof of an upper bound given in [11] on the number of polynomials f of degree d agreeing with t out of n distinct points $\{(x_i, y_i)\}$. Their proof uses an inclusion-exclusion argument, while ours is different. Notice that their bound is exactly the same and applies under exactly the same conditions on n, t and d, with the important difference being that our proof holds only for the univariate polynomial case, while theirs applies more generally. Nevertheless we feel that this new proof may be of some interest. Furthermore this justifies the statement, made in Section 1, that our algorithm works in exactly the same setting as the inclusion-exclusion bound.

Lemma 6 *If* $t/n \geq \left(\sqrt{2 + \frac{d}{4n}} \cdot \sqrt{\frac{d}{n}}\right) - \frac{d}{2n}$, *then the number of polynomials* f *of degree* d *satisfying (1) is at most*

$$\lfloor \frac{t}{d} + \frac{1}{2} - \sqrt{(\frac{t}{d} + \frac{1}{2})^2 - \frac{2n}{d}} \rfloor \leq \frac{2n}{t + d/2}.$$

Remark: The bound above shows that when t is linear in n, then the number of polynomials is bounded by a constant (see also Table 1). In particular, if $t < \frac{n+d}{2}$, (i.e., the number of errors is less than the error-correction limit), then the bound shown above is 1.

Proof: If m and l are integers such that the algorithm of the previous section works correctly, then the algorithm of Section 2 gives at most l solutions implying that l is an upper bound on the number of polynomials of degree d which can agree with n given points at t places. In other words, if m and l are integers satisfying

$$(m+1)(l+1) + d\binom{l+1}{2} > n. \tag{3}$$

$$\text{and } m + ld + 1 \le t, \tag{4}$$

then l is an upper bound on the number of functions satisfying (1). (4) indicates that we should pick m to be as large as possible subject to the constraint $m \le t - ld - 1$. We therefore set m to $t - ld - 1$. Thus (3) reduces to

$$(t - ld)(l+1) + d(l+1)l/2 > n.$$

In other words we require

$$(\frac{d}{2})\, l^2 - (t - \frac{d}{2})\, l + (n - t) < 0. \tag{5}$$

Let

$$\alpha \stackrel{\text{def}}{=} \frac{1}{d}\left((t - d/2) - \sqrt{(t-d/2)^2 - 4(d/2)(n-t)}\right)$$

and

$$\delta \stackrel{\text{def}}{=} \frac{2}{d}\left(\sqrt{(t-d/2)^2 - 4(d/2)(n-t)}\right).$$

Then α and $\alpha + \delta$ are the two roots to (5). If $\delta > 1$, then $l = \lfloor \alpha + 1 \rfloor$[1] and $m = t - ld - 1$, satisfy conditions above and l provides an upper bound on the number of functions satisfying (1).

The condition $\delta > 1$ is satisfied if

$$\sqrt{\left(\frac{2t}{d} - 1\right)^2 - 8\frac{n}{d} + 8\frac{t}{d}} > 1.$$

$$\Leftarrow \sqrt{\left(\frac{2t}{d} + 1\right)^2 - 8\frac{n}{d}} > 1.$$

[1] Notice that we need l to be an integer which satisfies the inequality (5) strictly and hence we are forced to use $\lfloor \alpha + 1 \rfloor$ rather than just $\lceil \alpha \rceil$.

$$\Leftarrow \left(\frac{2t}{d}+1\right)^2 - 8\frac{n}{d} > 1.$$
$$\Leftarrow \frac{2t}{d}+1 > \sqrt{8\frac{n}{d}+1}.$$
$$\Leftarrow t > \frac{d}{2}\left(-1+\sqrt{8\frac{n}{d}+1}\right) = n\left(\sqrt{\frac{d}{n}}\sqrt{2+\frac{d}{4n}} - \frac{d}{2n}\right).$$

If t satisfies the condition above then we get the following bound for l.

$$\begin{aligned} l &= \lfloor \alpha + 1 \rfloor \\ &= \lfloor (\frac{t}{d} - \frac{1}{2}) - \sqrt{(\frac{t}{d} - \frac{1}{2})^2 - \frac{2(n-t)}{d}} + 1 \rfloor \\ &= \lfloor (\frac{t}{d} + \frac{1}{2}) - \sqrt{(\frac{t}{d} + \frac{1}{2})^2 - \frac{2n}{d}} \rfloor. \end{aligned}$$

This yields the lemma. ([11] already show that the final quantity is upper bounded by $\frac{2n}{t+d/2}$, so we don't have to show that part.) ▮

5 Extensions to multivariate polynomials

It is relatively simple to extend the algorithm and the analysis of the earlier sections directly so as to apply for multivariate polynomial fitting.

We first extend the problem definition. Some slight care has to be take to determine what is an appropriate extension of the problem definition, and the definition below turns out to be the one for which the extension works easily.

We consider k-variate polynomials. We use $x^{(1)}, \ldots, x^{(k)}$ to denote the k variables. The shorthand $\hat{x}$ will be used to denote this tuple of variables in vector notation.

Problem 2
Input: *A field F, a set $H \subset F$, a function $f' : H^k \to F$ and integers t and d.*

Output: *A list of all functions $f : F^k \to F$ satisfying*

$$\left.\begin{array}{l} f(\hat{x}) \textit{ is a polynomial in } \hat{x} \textit{ of degree at most } d \\ \textit{and } |\{\hat{x} \in H^k | f(\hat{x}) = f'(\hat{x})\}| \geq t. \end{array}\right\} \quad (6)$$

The algorithm is a straightforward generalization of the algorithm of Section 2. We first extend the definition of weighted degree in the obvious way.

Definition 7 *For integers $w_1, \ldots, w_n$, the $(w_1, \ldots, w_n)$-weighted degree of a monomial $\prod_{i=1}^n x_i^{d_i}$ is $\sum_i w_i d_i$. The $(w_1, \ldots, w_n)$-weighted degree of a polynomial $Q(x_1, \ldots, x_n)$ is the maximum, over the monomials with non-zero coefficients in Q, of the $(w_1, \ldots, w_n)$ weighted degree of the monomial.*

Multivariate Algorithm: /* Inputs: d, t, k; $H \subset F$; $\{f'(x)\}_{x \in H^k}$ */

A′. /* Parameters l, m to be set later. */

B′. Find *any* function $Q : F^{k+1} \to F$ satisfying

$$\left.\begin{array}{l} Q(\hat{x}, y) \text{ has } (1, \ldots, 1, d)\text{-weighted degree at most } m + ld, \\ \forall \hat{x} \in H^k,\ Q(\hat{x}, f'(\hat{x})) = 0, \\ Q \text{ is not identically zero.} \end{array}\right\} \tag{7}$$

C′. Factor the polynomial Q into irreducible factors.

D′. Output all degree d polynomials f such that $(y - f(\hat{x}))$ is a factor of Q and $f(\hat{x}) = f'(\hat{x})$ for at least t values of $\hat{x} \in H^k$.

As usual we need to ensure that the number of coefficients of Q is more than n and prove that for sufficiently large t, the algorithm will output all solutions to the reconstruction problem above.

Claim 8 *If $m + ld \geq (k+1)(d+1)^{\frac{1}{k+1}} h^{\frac{k}{k+1}}$, then there exists a function $Q(\hat{x}, y)$ satisfying (7).*

Proof [Sketch]: The number of coefficients in Q is strictly larger than $\frac{1}{m+d+1}\binom{m+ld+k+1}{k+1}$. (The bound stated is a gross approximation. A generic degree $m + ld$ polynomial in $k+1$ variables has $\binom{m+ld+k+1}{k+1}$ coefficients. Q is a degree $m + ld$ polynomial in $k+1$ variables with the restriction that the degree of y only ranges from 0 to l. This restriction is taken care of by the $\frac{1}{m+d+1}$ factor.) Again this number can be lower bounded (grossly) by $\frac{1}{m+d+1}(\frac{m+ld+k+1}{k+1})^{k+1}$ and we wish this to be at least as large as $n = h^k$, which follows easily. ▌

Now it remains to mimic Lemma 4 in the multivariate setting.

Claim 9 *If $Q(\hat{x}, y)$ is a function satisfying (7) and $f(\hat{x})$ is a function satisfying (6) and $t > (m + ld)h^{k-1}$, then $(y - f(\hat{x}))$ divides $Q(\hat{x}, y)$.*

Proof [Sketch]: Proof similar to that of Lemma 4. The function $p(\hat{x}) \stackrel{\text{def}}{=} Q(\hat{x}, f(\hat{x}))$ is a polynomial of degree $m + ld$ and identically zero. Hence $f(\hat{x})$ is a root of the polynomial $Q_{\hat{x}}(y) \stackrel{\text{def}}{=} Q(\hat{x}, y)$. ▌

Thus with some optimization of parameters we are done.

Theorem 10 *Given a table for a function $f' : H^k \to F$, where F is a field and H is an arbitrary finite subset of F, and integers t and d, a list of all polynomials $f : F^k \to F$ of degree at most d which agree with f' in t places can be found in time* poly$(|H|^k)$ *provided*

$$\epsilon \stackrel{\text{def}}{=} \frac{t}{|H|^k} > (k+1)\left(\frac{d+1}{|H|}\right)^{\frac{1}{k+1}}.$$

Proof: Let $h = |H|$. Set $m = 0$ and $l = \frac{1}{d}(k+1)(d+1)^{\frac{1}{k+1}} h^{\frac{k}{k+1}}$. Then since $m + ld \geq (k+1)(d+1)^{\frac{1}{k+1}} h^{\frac{k}{k+1}}$, Lemma 8 guarantees that Step B′ of the multivariate algorithm will return a polynomial Q satisfying (7). Further the condition on t implies that $t \geq (m + ld)h^{k-1}$ indicating that if a function f satisfies (6), then $y - f(\hat{x})$ will divide Q. Thus all polynomials will be returned by this procedure.

6 Conclusions

We first discuss the univariate reconstruction algorithm. The limit on t, i.e., $t \gtrapprox \sqrt{2nd}$ is a significant weakness in practical usability of this code. Many

rate $\stackrel{\text{def}}{=} \frac{k}{n}$	error-correction bound	error tolerated by our algorithm	# solns reported by our algorithm
.5	.25	.25	1
.33	.333	.35	2
.25	.375	.417	2
.10	.45	.602	4

Table 1:

applications tend to work with $[n, k, \delta]$ codes where $\delta/n \to 0$, and for such growth our proof does not yield anything interesting. However low-rate codes, and even Hadamard codes, have been used in the past[2] and when it suffices to use a low-rate code the reconstruction algorithm has some advantage. In Table 1, we list the error which the reconstruction algorithm can tolerate and the number of solutions produced for such rates. The τ-reconstruction problem remains open for values of τ closer to the distance of the code, i.e., for $\tau = (1 - \epsilon)\delta$ where $\epsilon > 0$, and finding such a solution may be of both theoretical and practical interest. One does not expect the running time of such an algorithm to be polynomial in $\frac{1}{\epsilon}$, since Goldreich et al. [11] give a NP-hardness proof in this case. The NP-hardness result there has $n = 2d+3$, $t = d + 2$, with d being the degree of the desired polynomial. The instance is defined over the reals/rationals.

One of the main hopes in investigating this problem was that the algorithmic solution will be of some use in complexity theory. It is in this area that numerous application for "Reed Solomon like" codes have occurred repeatedly. Examples include (1) In determining the hardness of the permanent on random instances [16, 7], (2) Fault tolerant computing in distributed computing environments [3], and (3) Many result involving "probabilistically-checkable proofs". (See survey by Feigenbaum [8] for a detailed look at many connections.) In particular, in the case of applications to probabilistically checkable proofs, it becomes useful to be able to characterize functions that are not close to polynomials (to be able to refute claimed proofs of incorrect statements). Notice that our algorithm proves that the problem of recognizing

[2] Apparently the Mariner used a Hadamard code ([18], page 26).

points which are not very close to *any* polynomial is decidable in NP (by showing it is in P). The witness for this property is a proof that a function does not have a low-degree polynomial describing it on even a tiny fraction of the input. Recent work by Arora and Sudan [2] seems to justify our initial hope by using some of this analysis in a new analysis of low-degree testing, which in turn leads to new constructions of proof systems. Another application has also been shown (very recently) by Impagliazzo and Nisan [12] in the area of cryptography. They use the algorithm presented here to obtain "random-efficient constructions of hardcore predicates for one-way functions". (Defining any of these terms is beyond the scope of this paper.)

Now, moving on to the multivariate reconstruction problem, several glaring open problems remain. For starters, it is not clear as to how to even bound the number of solutions when $\epsilon \stackrel{\text{def}}{=} \frac{t}{|H|^k}$ grows slower than $o(\sqrt{d/|H|})$. Even in the cases where ϵ is larger than $\Omega(\sqrt{d/|H|})$, the reconstruction problem is not fully solved. In particular even for the case $k = 2$, no algorithm appears to solve this problem efficiently. This should hopefully be some oversight on our part and some simple modification of methods in [9, 1, 11] may work. However such a solution does not remain algebraic. The question of whether there is an algebraic solution to the k-variate problem for general k, which works whenever $\epsilon = \Omega(\sqrt{d/|H|})$, seems to be another interesting question.

Lastly we speculate on the complexity of the maximum-likelihood problem (or the nearest codeword problem). This problem is known to be NP-hard for general linear codes [5]. The hardness of the problem considered in [5] could be due to one of two reasons: (1) It is a maximum likelihood decoding problem rather than a τ-error correction problem; (2) The code is specified as part of the input, rather than being a "well-known" one which is more standard. It would be nice to know which of the two causes is responsible for the hardness, since for all well-known codes, the τ-error correction problem seems to well-solved. Bruck and Naor [6] present a code (not well-known, but nevertheless easily presented), for which they show that the existence of small size maximum likelihood decoding circuits would imply the collapse of the polynomial hierarchy (using a result of Karp and Lipton's [14]). However the codes presented by Bruck and Naor do not have large distance. It still remains open if the maximum likelihood decoding problem is hard for any constant distance code. The Reed Solomon codes would have formed a good candidate to show hardness of this problem, except that it is not so hard to solve. It would be nice to find some other candidate for such a hardness result. Lastly, there is still the possibility that some error-correcting codes are hard to decode, to even the full extent of their error-correction capacity. Alternately, we can ask the question: Is it possible to construct a τ-error correction algorithm for every τ-error correcting linear code, specified by its generator matrix? A positive answer to this might necessitate an algorithm to determine the distance of a code, a well-known open problem.

Acknowledgments

I am grateful to Sanjeev Arora, Oded Goldreich, Ronitt Rubinfeld, Martin Tompa and Shmuel Winograd for many valuable comments and discussions. I am especially grateful to Greg Sorkin for spending numerous hours implementing a version of the algorithm presented here.

References

[1] S. Ar, R. Lipton, R. Rubinfeld and M. Sudan. Reconstructing algebraic functions from mixed data. *Proceedings of the 33rd Annual IEEE Symposium on Foundations of Computer Science*, pp. 503–512, 1992.

[2] S. Arora and M. Sudan. Manuscript, August 1996.

[3] M. Ben-Or, S. Goldwasser and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. *Proceedings of the 20th Annual ACM Symposium on Theory of of Computing*, pp. 1–10, 1988.

[4] E. Berlekamp and L. Welch. Error Correction of Algebraic Block Codes. *US Patent*, Number 4,633,470, 1986.

[5] E. Berlekamp, R. McEliece and H. van Tilborg. On the inherent intractability of certain coding problems. *IEEE Transactions on Information Theory*, pp. 384–386, 1978.

[6] J. Bruck and M. Naor. The hardness of decoding linear codes with preprocessing. *IEEE Transactions on Information Theory*, pp. 381–385, 1990.

[7] U. Feige and C. Lund. On the hardness of computing the permanent of random matrices. *Proceedings of the 24th Annual ACM Symposium on Theory of of Computing*, pp. 643–654, 1992.

[8] J. Feigenbaum. The Use of Coding Theory in Computational Complexity. *Proceedings of Symposia in Applied Mathematics*, R. Calderbank (ed.), American Math Society, Providence, pp. 203–229, 1995.

[9] P. Gemmell and M. Sudan. Highly resilient correctors for polynomials. *Information Processing Letters*, 43 (1992), 169–174.

[10] O. Goldreich and L.A. Levin. A Hard-Core Predicate for any One-Way Function. *Proceedings of the 21st Annual ACM Symposium on Theory of of Computing*, pp. 25–32, 1989.

[11] O. Goldreich, R. Rubinfeld and M. Sudan. Learning polynomials with queries: The highly noisy case. Full version in preparation. Extended abstract appears in *Proceedings of the 36th Annual IEEE Symposium on Foundations of Computer Science*, pp. 294–303, 1995.

[12] R. Impagliazzo and N. Nisan. Personal Communication. September 1996.

[13] E. Kaltofen. Polynomial Factorization 1987–1991. *LATIN '92*, I. Simon (Ed.), Springer LNCS, vol. 583, pp. 294–313, 1992.

[14] R. Karp and R. Lipton. Some connections between nonuniform and uniform complexity classes. *Proceedings of the 12th Annual ACM Symposium on Theory of of Computing*, pp. 302–309, 1980.

[15] E. Kushilevitz, Y. Mansour. Learning decision trees using the Fourier spectrum. *Proceedings of the 23th Annual ACM Symposium on Theory of of Computing*, pp. 455–464, 1991.

[16] R. Lipton. New directions in testing. *Distributed computing and cryptography, DIMACS Series in Discrete Math. and Th. CS*, vol. 2, AMS, 1991.

[17] F. MacWilliams and N. Sloane. *The Theory of Error-Correcting Codes.* North-Holland, 1981.

[18] J. H. van Lint. *Introduction to Coding Theory.* Springer-Verlag, 1982.