Ideal Error-correcting codes: Unifying Algebraic and Number-theoretic Algorithms

Madhu Sudan *

MIT Laboratory for Computer Science,
200 Technology Square,
Cambridge, MA 02139, USA.
madhu@mit.edu,
http://theory.lcs.mit.edu/~madhu

Abstract. Over the past five years a number of algorithms decoding some well-studied error-correcting codes far beyond their "error-correcting radii" have been developed. These algorithms, usually termed as list-decoding algorithms, originated with a list-decoder for Reed-Solomon codes [36, 17], and were soon extended to decoders for Algebraic Geometry codes [33, 17] and as also some number-theoretic codes [12, 6, 16]. In addition to their enhanced decoding capability, these algorithms enjoy the benefit of being conceptually simple, fairly general [16], and are capable of exploiting soft-decision information in algebraic decoding [24]. This article surveys these algorithms and highlights some of these features.

1 Introduction

List-decoding was introduced in the late fifties by Elias [7] and Wozencraft [38]. Under this model, a decoder is allowed to output a *list* of possible codewords that a corrupted received word may correspond to. Decoding is considered successful if the transmitted word is included in this list of received words.

While the initial model was introduced to refine the study of probabilistic channels, it has slowly developed into a tool for improving our understanding of error-correction even in adversarial models of error. Strong combinatorial results are known that bound the "list-decoding radius" of an error-correcting code as a function of its rate and its distance (See [5, 8, 40] for some of the earlier results, and [13, 15, 21] for some recent progress.) However till the late 90's no non-trivial algorithms were developed to perform efficient list-decoding. In [36], the author gave an algorithm to list-decode Reed-Solomon codes. This was shortly followed up by an algorithm by Shokrollahi and Wasserman [33] to decode algebraic-geometry codes. Subsequently the algorithms have been extended to decode many families of codes. Furthermore the efficiency of the original algorithms has been vastly improved and many applications have been found for this concept.

* Parts of this work were supported by NSF Grant CCR 9875511, NSF Grant CCR 9912342, and an Alfred P. Sloan Foundation Fellowship.

In this paper we describe the basic ideas behind the decoding algorithms. Our focus is mostly on the simplicity of these algorithms and not so much on their performance or uses.

2 Reed-Solomon Decoding

Let $\mathbb{F}_q$ denote a field of size q and let $\mathbb{F}_d^k[x]$ denote the vector space of polynomial of degree at most k over $\mathbb{F}_q$. Recall that the Generalized Reed Solomon code of dimension k, is specified by distinct $x_1, \ldots, x_n \in \mathbb{F}_q$ and consists of the evaluations of all polynomials p of degree at most k at the points $x_1, \ldots, x_n$. More formally, letting $\boldsymbol{x} = \langle x_1, \ldots, x_n \rangle$ and letting $p(\boldsymbol{x})$ denote $\langle p(x_1), \ldots, p(x_n) \rangle$, we get that the associated code $\mathrm{RS}_{q,k,\boldsymbol{x}}$ is given by

$$\mathrm{RS}_{q,k,\boldsymbol{x}} = \{p(\boldsymbol{x}) | p \in \mathbb{F}_q^k[x]\}.$$

Viewed from this perspective (as opposed to the dual perspective, where the codewords of the Reed Solomon codes are coefficients of polynomials), the Reed Solomon decoding problem is really a "curve-fitting" problem: Given n-dimensional vectors $\boldsymbol{x}$ and $\boldsymbol{y}$, find all polynomial $p \in \mathbb{F}_q^k[x]$ such that $\Delta(p(\boldsymbol{x}), \boldsymbol{y}) \leq e$. for some error parameter e. (Here and later $\Delta(\cdot, \cdot)$ denotes the Hamming distance.)

Traditional algorithms, starting with those of Peterson [30] attempt to "explain" $\boldsymbol{y}$ as a function of $\boldsymbol{x}$. This part becomes explicit in the work of Welch & Berlekamp [37, 3] (see, in particular, the exposition in [35, Appendix A]) where $\boldsymbol{y}$ is interpolated as a rational function of $\boldsymbol{x}$, and this leads to the efficient decoding. (Specifically a rational function $a(x)/b(x)$ can be computed such that for all $i \in \{1, \ldots, n\}$,. $a(x_i) = y_i * b(x_i)$.)

Rational functions, however, are limited in their ability to explain data with large amounts of error. To motivate this point, let us consider the following simple (and contrived) channel: The input and output alphabet of the channel are $\mathbb{F}_q$. The channel behavior is as follows: On input a symbol $\alpha \in \mathbb{F}_q$, the channel outputs α with probability $\frac{1}{2}$ and $\omega\alpha$ with probability $\frac{1}{2}$, for some fixed $\omega \in \mathbb{F}_q$. Now this is a channel that makes an error with probability $\frac{1}{2}$, but still the information it outputs is very closely correlated with the input (and its capacity, in the sense of Shannon, is very close to 1). However if we transmit a Reed Solomon codeword on this channel, the typical output vector $\boldsymbol{y}$ does not admit a simple description as a rational function of $\boldsymbol{x}$, and thus traditional decoding algorithms fail.

However it is clear that the output of the channel is explain by some nice algebraic relations: Specifically, there exists a polynomial p (of degree at most k) such that for every i, $y_i = p(x_i)$ or $y_i = \omega \cdot p(x_i)$. The "Or" of two Boolean conditions also has a simple algebraic representation: we simply have that the polynomial $Q(x, y) = (y - p(x)) \cdot (y - \omega \cdot p(x))$ is zero on every given (x_i, y_i). Furthermore such a polynomial $Q(x, y)$ can be found by simple interpolation (which amounts to solving a linear system), and the candidate polynomial $p(x)$ can be determined as a root of the polynomial $Q(x, y)$. (Notice that the factoring

will find two polynomials p_1 and p_2 and, if $\omega^2 \neq 1$, the true candidate is p_1 iff it satisfies $p_2 = \omega p_1$.)

The above example illustrates the power of using algebraic curves (over rational functions) in decoding Reed-Solomon codes. The idea of using such functions was proposed by Ar et al. [1] who showed that if, by some fortunate occurrence, the vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ could be explained by some nice algebraic relation, then decomposing the algebraic curve (i.e., factoring) could tell if there exist large subsets of the data that satisfy non-trivial algebraic correlation. However they could not show general conditions under which the vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ could be explained by a nice algebraic curve, and this prevented them from obtaining a general decoding algorithm for Reed Solomon codes.

The complementing result took a few years to emerge, and did so finally in [36], where a simple counting argument is used to show that any pair of vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ has a "nice" algebraic curve explaining it. The x- and y-degree of the curve can be chosen as desired, subject to the condition that the support has at least $n+1$ coefficients. Putting these two pieces together, and choosing x- and y- degrees appropriately, one obtains the following algorithm and result:

Definition 1. *Let (w_x, w_y)-weighted degree of a monomial $x^i y^j$ be $i \cdot w_x + j \cdot w_y$. The (w_x, w_y)-weighted degree of a polynomial $Q(x, y)$ is the maximum, over all monomials with non-zero coefficient in Q, of their (w_x, w_y)-weighted degree.*

Given $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{F}_q^n$ and k.

1. Compute $Q \neq 0$ with $(1, k)$-weighted degree at most $\lfloor \sqrt{2(k-1)n} \rfloor$ satisfying $Q(x_i, y_i) = 0$ for all $i \in \{1, \ldots, n\}$ (this is simple interpolation).
2. Factor Q and report all polynomials $p \in \mathbb{F}_q^k[x]$ such that $y - p(x)$ is a factor of Q and $p(x_i) = y_i$ for $\lfloor \sqrt{2kn} \rfloor + 1$ values of $i \in \{1, \ldots, n\}$.

Theorem 1 ([36]). *Given vectors $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{F}_q^n$, a list of all polynomials $p \in \mathbb{F}_q^k[x]$ satisfying $p(x_i) = y_i$ for more than $\sqrt{2nk}$ values of $i \in \{1, \ldots, n\}$ can be found in time polynomial in n, provided all pairs (x_i, y_i) are distinct.*

The interesting aspect of the above algorithm is that it takes some very elementary algebraic concepts, such as unique factorization, Bezout's theorem, and interpolation, and makes algorithmic use of these concepts in developing a decoding algorithm for an algebraic code. This may also be a good point to mention some of the significant advances made in the complexity of factoring multivariate polynomials that were made in the 1980's. These algorithms, discovered independently by Grigoriev [14], Kaltofen [22], and Lenstra [25], form the technical foundations of the decoding algorithm above. Modulo these algorithms, the decoding algorithm and its proof rely only on elementary algebraic concepts. Exploiting slightly more sophisticated concepts from commutative algebra, leads to even stronger decoding results that we describe next.

The algorithm of Guruswami and Sudan [17] is best motivated by the following weighted curve fitting question: Suppose in addition to vectors $\boldsymbol{x}$ and $\boldsymbol{y}$, one is also given a vector of positive integers $\boldsymbol{w}$ where w_i determines the "weight"

or confidence associated with a given point (x_i, y_i). Specifically we would like to find all polynomials p such that $\sum_{i|p(x_i)=y_i} w_i \geq W$ (for as small a W as possible).

The only prior algorithm (known to this author) that could take such "reliability" consideration into account was the Generalized Minimum Distance (GMD) decoding algorithm of Forney [10]. This algorithm, in combination with Theorem 1, can find such a vector provided $W = \Omega\left(\sqrt{k}\sqrt{\ln n}\sqrt{\sum_{i=1}^{n} w_i^2}\right)$. However, the GMD algorithm is combinatorial, and we would like to look for a more algebraic solution.

How can one interpret the weights in the algebraic setting? A natural way at this stage is to find a "fit" for all the data points that corresponds to the weights: Specifically, find a polynomial $Q(x, y)$ that "passes" through the point (x_i, y_i) at least w_i times. The notion of a curve passing through a point multiple times is a well-studied one. Such points are called singularities. Over fields of characteristic zero, these are algebraically characterized by the fact that the partial derivatives of the curve (all such, upto the $(r-1)$th derivatives, if the point must be visited by the curve r times), vanish at the point. The relevant component of this observation is that insisting that a curve pass through a point r times is placing $\binom{r}{2}$ linear constraints on the coefficients. This fact remains true over finite fields, though the partial derivatives don't yield these linear constraints any more. Using this notion to find curves that fit the points according to the weights, and then factoring the curves, leads to the following algorithm and result.

Given $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{F}_q^n$, $\boldsymbol{w} \in \mathbb{Z}_{\geq 0}^n$, and k.

1. Compute $Q \neq 0$ with $(1, k)$-weighted degree at most $\lfloor \sqrt{k \sum_{i=1}^{n} w_i(w_i+1)} \rfloor$ satisfying $Q(x_i, y_i)$ is a zero of multiplicity w_i, for all $i \in \{1, \ldots, n\}$.
2. Factor Q and report all polynomials $p \in \mathbb{F}_q^k[x]$ such that $y - p(x)$ is a factor of Q and $\sum_{i|p(x_i)=y_i} w_i$ is at least $\lfloor \sqrt{k \sum_{i=1}^{n} w_i(w_i+1)} \rfloor + 1$.

Lemma 1 ([17]). *Given vectors $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{F}_q^n$, a list of all polynomials $p \in \mathbb{F}_q^k[x]$ satisfying $\sum_{||p(x_i)=y_i} w_i > \lfloor \sqrt{k \sum_{i=1}^{n} w_i(w_i+1)} \rfloor$ can be found in time polynomial in $n, \sum_i w_i$, provided all pairs (x_i, y_i) are distinct.*

At first glance it is not clear if this is better than the GMD bound. The GMD bound is invariant with respect to scaling of the w_i's while the above is not! In fact, it is this aspect that makes the algorithm above intriguing. Fix vectors $\boldsymbol{x}$ and $\boldsymbol{y}$, and consider two possible weight assignments: in the first all weights are 1, and in the second all weights are 2. On the one hand, the weight vectors place the same relative weights on all points, so a "good" solution to the first instance is also a "good" solution to the second instance. On the other hand, a close examination of the bound in Lemma 1 reveals that in the latter case it can find some polynomials that the former can not. The first instance finds all polynomials that agree with the data in $\sqrt{2kn}$ points, while the second finds all polynomials that agree with the data in $\sqrt{\frac{3}{2}kn}$ points. Scaling the weights to larger and larger values, in the limit we find all polynomials that fit the data

over more than $\sqrt{kn}$ points. The price we pay is that the running time of the algorithm grows with the scaling factor. However it is easy to see that a finite (polynomial in n) weight suffices to decode up to this bound and this leads to the following theorem:

Theorem 2 ([17]). *Given vectors $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{F}_q^n$, a list of all polynomials $p \in \mathbb{F}_q^k[x]$ satisfying $p(x_i) = y_i$ for more than $\sqrt{nk}$ values of $i \in \{1, \ldots, n\}$ can be found in time polynomial in n, provided all pairs (x_i, y_i) are distinct.*

Note that while the original motivation was to find a better algorithm for the "weighted" decoding problem, the result is a better unweighted decoding algorithm, that uses the weighted version as an intermediate step. Of course, it is also possible to state what the algorithm achieves for a general set of weights. For this part, we will just assume that the weight vector is an artbitrary vector of non-negative reals, and get the following:

Theorem 3 ([17, 18]). *Given vectors $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{F}_q^n$, a weight vector $\boldsymbol{w} \in \mathbb{R}_{\geq 0}^n$, and a real number $\epsilon > 0$, a list of all polynomials $p \in \mathbb{F}_q^k[x]$ satisfying $\sum_{i|p(x_i)=y_i} w_i > \sqrt{k(\epsilon + \sum_{i=1}^n w_i^2)}$ can be found in time polynomial in n and $\frac{1}{\epsilon}$, provided the pairs (x_i, y_i) are all distinct.*

This result summarizes the state of knowledge for list-decoding for Reed Solomon codes, subject to the restriction that the decoding algorithm runs in polynomial time. However this criterion, that the decoding algorithm runs in polynomial time, is a very loose one. The practical nature of the problem deserves a closer look at the components involved and efficient strategies to implement these components. This problem has been considered in the literature, with significant success. In particular, it is now known how to implement the interpolation step in $O(n^2)$ time, when the output list size is a constant [29, 31]. Similar running times are also known for the root finding problem (which suffices for the second step in the algorithms above) [2, 11, 28, 29, 31, 39]. Together these algorithms lead to the possibility that a good implementation of list-decoding may actually even be able to compete with the classical Berkelkamp-Massey decoding algorithm in terms of efficiency. A practical implementation of such an algorithm in C++, due to Rasmus Refslund Nielsen, is available from from his homepage (`http://www.student.dtu.dk/~p938546/index.html`).

3 Ideal Error-correcting codes and decoding

We now move on to other list-decoding algorithms for other algebraic codes. The potential for generalizing the decoding algorithms above to codes other than just the Reed Solomon code, was first shown by Shokrollahi and Wasserman [33]. In their work, they show how to generalize the algorithm above to decode the more general family of algebraic-geometry codes. A full description of this family of codes is out of scope for this article — the reader is encouraged to read the text of Stichtenoth [34] or the article by Høholdt, van Lint, and Pellikaan [20] for

a description. However we will attempt to describe the flavor of the results by defining a broad class of codes, that we call "Ideal error-correcting codes".

One way of viewing Reed Solomon codes, is that they are built over a (nice) integral domain $R = \mathbb{F}_q[x]$, [1] The message space $\mathcal{M} = \mathbb{F}_q^k[x]$ is chosen to be a subset of the ring R. Additionally the code is specified by a collection of ideals $I_1, \ldots, I_n$ of R. In the case of Reed Solomon codes, these are the ideals generated by the linear polynomials $x - x_1, \ldots, x - x_n$. The encoding of a message element $p \in R$ is simply its residue modulo n ideals. Thus, in Reed Solomon encoding, $p \mapsto \langle p \bmod (x - x_1), \ldots, p \bmod (x - x_n)\rangle = p(\boldsymbol{x})$ as expected. The following definition summarizes the family of codes obtained this way.

Definition 2 (Ideal error-correcting codes [16]). *An ideal error-correcting code is specified by a triple $(R, \mathcal{M}, \langle I_1, \ldots, I_n\rangle)$, where R is an integral domain, $\mathcal{M} \subseteq R$, and $I_1, \ldots, I_n$ are ideals of R. The code is a subset of $(R/I_1) \times \cdots \times (R/I_n)$, given by the set $\{\langle p \bmod (I_1), \ldots, p \bmod (I_n)\rangle | p \in \mathcal{M}\}$.*

To quantify the distance properties of such a code, it is useful to impose a notion of size on elements of the ring R. In the case of Reed Solomon codes the size of an element is essentially its degree (though for technical reasons, it is convenient to use $q^{\deg p}$ as a measure of size). The message space usually consists of all elements of small size. To make this space large one needs to know that the ring has sufficiently many small elements. Further the size function is assumed to satisfy some axioms such as $\text{size}(a+b) \leq \text{size}(a) + \text{size}(b)$, $\text{size}(ab) \leq \text{size}(a) \cdot \text{size}(b)$ and so on. Further, if the size of an ideal is defined to be the size of the smallest non-zero element in it, then $\text{size}(J_1 \times J_2)$ should be at least $\text{size}(J_1) \cdot \text{size}(J_2)$. Assuming such, relatively simple axioms it is possible to analyze the minimum distance of an ideal error-correcting code, once the sizes of the ideals I_1 to I_n are known. (We will not cover these definitions formally here - we refer the reader to [16] for a full discussion.) The same axioms guarantee efficient (list-)decoding as well. In fact, the following simple generalization of the algorithm from the previous section gives the algorithm for decoding any ideal error-correcting code. We describe the algorithm informally. Formal specification will involve a careful setting to various parameters.

Given $\boldsymbol{I}$ $\boldsymbol{y} \in R^n$, $\boldsymbol{w} \in \mathbb{Z}_{\geq 0}^n$.

1. Let $J_i = I_i + (y - y_i)$.
2. Compute $Q \in R[y] - \{0\}$ of small degree in y, with small coefficients, satisfying $Q \in \prod_{i=1}^n J_i^{w_i}$.
3. Factor Q and report all elements $p \in \mathcal{M}$ such that $y - p$ is a factor of Q and $y_i \in p + (I_i)$ for sufficiently many i.

[1] For the reader that is rusty with the elements of commutative algebra, let us recall that an integral domain is a commutative ring R that has no zero divisors (i.e., $pq = 0$ implies $p = 0$ or $q = 0$). An ideal I in R is a subset that is closed under addition, and $a \in I$ implies $ab \in I$, for all $b \in R$. The quotient of R over I, denoted R/I forms an integral domain and this quotient ring is crucial to many definitions here.

In this setting, the algorithm above may even appear more natural. Note that the ideals J_i above have the following meaning: $y - p$ belongs to the ideal J_i if and only if $y_i \in p + I_i$. Thus we want all elements p such that $y - p$ lies in many of the ideals J_i. To find such an element, we find an element that Q that lies in all of them, and factor it to find any element that lies in many of them.

Why consider this more complicated scheme? Ideal error-correcting codes not only include the class of Reed Solomon codes (as already pointed out), but also all algebraic-geometry codes, and an interesting family of number-theoretic codes termed Redundant Residue Number System (RRNS) codes. As a consequence of the generalization above, one gets a structure for decoding all the above family of codes. Note that we only get a structure, not the algorithm itself. In order to get actual decoding algorithms, one needs to find algorithms to "Compute Q" (the interpolation step) as well as to factor over $R[y]$. Both aspects present their own complexity, as we will illustrate for the RRNS codes. Furthermore, to get the best possible decoding algorithm, one needs to select the parameters, and in particular the weights carefully. We will discuss this more in the next section.

Finally, we point out one important class of codes where the decoding algorithms don't seem to apply. This is the class of Reed-Muller codes where the algorithm of Feng and Rao [9] (see, in particular, the desciption in [20]) decodes up to half the minimum distance. The best known list-decoding algorithm [32] does better than the above algorithm for some choice of parameters, but does not even match up to the above algorithm for other choices of parameters. Extending the list-decoding algorithm given here to apply to the class of Reed-Muller codes seems to require a generalization beyond the class of ideal codes.

4 Redundant residue number system codes

This is the family of ideal error-correcting codes given by $R = \mathbb{Z}$, $\mathcal{M} = \{0, \ldots, K-1\}$ for some integer K, and $I_i = (p_i)$ for a collection of pairwise prime integers $p_1, \ldots, p_n$. In other words a message is a non-negative integer less than K and its encoding are its residues modulo small integers $p_1, \ldots, p_n$. If we permute the indices so that $p_1 \leq \cdots \leq p_n$, and if $K \leq \prod_{i=1}^{k} p_i$, then this code has minimum distance at least $n - k + 1$, Thus it should be correctible to up to $\frac{n-k}{2}$ errors uniquely, and list-decodable to about $n - \sqrt{nk}$ errors. Turns out Mandelbaum [27] gave a unique decoding algorithm decoding to $\frac{n-k}{2}$ errors. The algorithm runs in polynomial time provided the highest and smallest moduli are relatively close in value. Goldreich et al. [12] gave an algorithm correcting approximately $n - \sqrt{2nk\frac{\log p_n}{\log p_1}}$ errors. Boneh [6] improved this to $n - \sqrt{nk\frac{\log p_n}{\log p_1}}$ errors, and finally Guruswami et al. [16] improved this to correct $n - \sqrt{n(k+\epsilon)}$ errors for arbitrarily small ϵ. They also give a polynomial time unique decoding algorithm correcting up to $\frac{n-k}{2}$ errors.

The algorithms of [12, 6, 16] illustrate some of technicalities that surface in specializing the algorithm of Section 3. For instance, consider the interpolation step: Even in the simple case when all the w_i's are 1, the case considered in [12],

the algorithm for this part is not obvious. We wish to find a polynomial Q with small integer coefficients such that $Q(y_i) = 0 \bmod (\prod_{i=1}^{n} p_i)$. This is a task in Diophantine approximation and no longer a simple linear system. Fortunately, it turns out to be a relatively well-studied problem. The set of polynomials satisfying the condition $Q(y_i) = 0 \bmod (\prod_{i=1}^{n} p_i)$ form a lattice, and finding a polynomial with small coefficients is a "shortest vector problem" in integer lattices and one can use the groundbreaking algorithm of Lenstra, Lenstra, and Lovasz [26] (LLL) to solve this problem near-optimally. In the case of general weights [6, 16], the problem remains a short vector problem in a lattice, however it is not simple to express a basis for this lattice explicitly. In the case of uniform, but not unit weights, [6] manages to come up with an explicit description based on some analogies with some problems in cryptography. For the fully general case, [16] do not describe an explicit basis. Instead they give an algorithm that computes this basis from the weights. Thus this step of the process can get quite complicated.in the case of numkber theoretic codes. (In contrast this step remains reasonably simple in the case of algebraic geometry codes.)

Another aspect of the decoding algorithm highlighted by the number-theoretic setting is the choice of weights. Even in the simple case where all the input weights are unit, it is not clear that the best choice of weights is a uniform one. Indeed, the final choice used by [16] gives large weights to the small moduli and smaller weights to the larger modulii. In general, this issue — what is the best choice of weights to the algorithm, and how should they relate to the weights given as input — is far from clear. For example, a recent paper of Kötter and Vardy [24] suggests a completely surprising choice of weights in the case of algebraic geometry codes. This leads to better bounds for decoding these codes that the one given in [17]. A more detailed examination of this question has been carried out by Kötter [23].

Finally, we move on to the second step of the decoding algorithm. In this case the algorithm that is required is an integer root-finding algorithm for integer polynomials. This is again a well-studied problem, with known polynomial time solutions. This step however can get significantly more complicated for other ideals. E.g., in the case of algebraic-geometry codes, the issue becomes that of how the codes are specified. For most well-known families of such codes, the standard specifications do lead to polynomial time solutions [11, 28, 29, 39]. For arbitrary codes, however it is a priori unclear if a natural representation could lead to a polynomial time decoding algorithm. In fact in the absence of a complete characterization of all algebraic geometry codes, it is unclear as to what is a natural representation for all of them. [19] suggest a potential representation that is reasonably succinct (polynomial sized in the generator matrix), that allows this and other necessary tasks to be solved in polynomial time, by using the algorithms of [11] and [29].

Acknowledgments. Thanks to Venkatesan Guruswami for letting me describe many of our joint works here, to Tom Høholdt for enlightening me on many of the developments (both old and new) in the coding theory community, and to Ralf Kötter for clarifying the subtleties in the choice of weights.

References

1. Sigal Ar, Richard J. Lipton, Ronitt Rubinfeld, and Madhu Sudan. Reconstructing algebraic functions from erroneous data. *SIAM Journal on Computing*, 28(2): 487–510, April 1999.
2. Daniel Augot and Lancelot Pecquet. A Hensel lifting to replace factorization in list decoding of algebraic-geometric and Reed-Solomon codes. *IEEE Trans. Info. Theory*, 46(6): 2605-2613, November 2000.
3. Elwyn R. Berlekamp. Bounded distance +1 soft-decision Reed Solomon decoding. *IEEE Transactions on Information Theory*, 42(3):704-720, 1996.
4. Richard E. Blahut. Theory and practice of error control codes. Addison-Wesley Pub. Co., 1983.
5. V. M. Blinovskii. Bounds for codes in the case of list decoding of finite volume. *Problemy Peradachi Informatsii*, 22(1):11–25, January-March 1986.
6. Dan Boneh. Finding smooth integers in short intervals using CRT decoding. (To appear) *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing*, Portland, Oregon, 21-23 May 2000.
7. Peter Elias. List decoding for noisy channels. *WESCON Convention Record*, Part 2, Institute of Radio Engineers (now IEEE), pages 94–104, 1957.
8. Peter Elias. Error-correcting codes for list decoding. *IEEE Transactions on Information Theory*, 37(1):5–12, January 1991.
9. G.-L. Feng and T. R. N. Rao. Decoding algebraic-geometric codes upto the designed minimum distance. *IEEE Transactions on Information Theory*, 39(1):37–45, January 1993.
10. G. David Forney Jr.. *Concatenated Codes.* MIT Press, Cambridge, MA, 1966.
11. S. Gao and M. A. Shokrollahi. Computing roots of polynomials over function fields of curves. *Proceedings of the Annapolis Conference on Number Theory, Coding Theory, and Cryptography*, 1999.
12. Oded Goldreich, Dana Ron, and Madhu Sudan. Chinese remaindering with errors. *IEEE Transactions on Information Theory.* 46(4): 1330–1338, July 2000.
13. Oded Goldreich, Ronitt Rubinfeld, and Madhu Sudan. Learning polynomials with queries: The highly noisy case. *Proceedings of the 36th Annual Symposium on Foundations of Computer Science*, pages 294–303, Milwaukee, Wisconsin, 23-25 October 1995.
14. Dima 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. Venkatesan Guruswami, Johan Håstad, Madhu Sudan, and David Zuckerman. Combinatorial bounds for list decoding. (To appear) *Proceedings of the 38th Annual Allerton Conference on Communication, Control, and Computing*, 2000.
16. Venkatesan Guruswami, Amit Sahai, and Madhu Sudan. "Soft-decision" decoding of Chinese remainder codes. *Proceedings of the 41st Annual Symposium on Foundations of Computer Science*, pages 159-168, Redondo Beach, California, 12-14 November, 2000.
17. Venkatesan Guruswami and Madhu Sudan. Improved decoding of Reed-Solomon codes and algebraic-geometric codes. *IEEE Transactions on Information Theory*, 45(6): 1757–1767, September 1999.
18. Venkatesan Guruswami and Madhu Sudan. List decoding algorithms for certain concatenated codes. Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing, pages 181-190, Portland, Oregon, 21-23 May 2000.

19. Venkatesan Guruswami and Madhu Sudan, On representations of algebraic-geometric codes. IEEE Transactions on Information Theory (to appear).
20. T. Høholdt, J. H. van Lint, and R. Pellikaan. Algebraic geometry codes. In *Handbook of Coding Theory*, V. Pless and C. Huffman (Eds.), Elsevier Sciences, 1998.
21. J. Justesen and T. Høholdt. Bounds on list decoding of MDS codes. Manuscript, 1999.
22. Erich 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.
23. Ralf Kötter. Personal communication, March 2001.
24. Ralf Kötter and Alexander Vardy. Algebraic soft-decision decoding of Reed-Solomon codes. (To appear) *Proceedings of the 38th Annual Allerton Conference on Communication, Control, and Computing*, 2000.
25. Arjen K. Lenstra. Factoring multivariate polynomials over finite fields. *Journal of Computer and System Sciences*, 30(2):235–248, April 1985.
26. A. K. Lenstra, H. W. Lenstra, and L. Lovasz. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261:515–534, 1982.
27. D. M. Mandelbaum. On a class of arithmetic codes and a decoding algorithm. *IEEE Transactions on Information Theory*, 21:85–88, 1976.
28. R. Matsumoto. On the second step in the Guruswami-Sudan list decoding algorithm for AG-codes. *Technical Report of IEICE*, pp. 65-70, 1999.
29. R. Refslund Nielsen and Tom Høholdt. Decoding Hermitian codes with Sudan's algorithm. *Proceedings of AAECC-13, LNCS 1719, Springer-Verlag*, 1999, pp. 260-270.
30. W. W. Peterson. Encoding and error-correction procedures for Bose-Chaudhuri codes. *IRE Transactions on Information Theory*, IT-60:459-470, 1960.
31. Ron M. Roth and Gitit Ruckenstein. Efficient decoding of Reed-Solomon codes beyond half the minimum distance. *IEEE Transactions on Information Theory*, 46(1):246-257, January 2000.
32. Madhu Sudan, Luca Trevisan, and Salil Vadhan. Pseudorandom generation without the XOR lemma. *Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing*, pages 537-546, Atlanta, Georgia, 1-4 May 1999.
33. M. Amin Shokrollahi and Hal Wasserman. List decoding of algebraic-geometric codes. *IEEE Transactions on Information Theory*, 45(2): 432-437, March 1999.
34. Henning Stichtenoth. *Algebraic Function Fields and Codes.* Springer-Verlag, Berlin, 1993.
35. Madhu Sudan. *Efficient Checking of Polynomials and Proofs and the Hardness of Approximations.* ACM Distinguished Theses. Lecture Notes in Computer Science, no. 1001, Springer, 1996.
36. Madhu Sudan. Decoding of Reed Solomon codes beyond the error-correction bound. *Journal of Complexity*, 13(1): 180–193, March 1997.
37. Lloyd Welch and Elwyn R. Berlekamp. Error correction of algebraic block codes. *US Patent* Number 4,633,470, issued December 1986.
38. J. M. Wozencraft. List decoding. Quarterly Progress Report. Research Laboratory of Electronics, MIT, Vol. 48, pp. 90–95, 1958.
39. Xin-Wen Wu and Paul H. Siegel. Efficient list decoding of algebraic geometric codes beyond the error correction bound. *Proc. of International Symposium on Information Theory*, June 2000.
40. V. V. Zyablov and M. S. Pinsker. List cascade decoding. *Problemy Peredachi Informatsii*, 17(4):29–33, October-December 1981.