List Decoding

Thomas Grubb

May 3, 2015

1 Introduction

In sending messages or storing data, it is often the case that information will need to be encoded, transmitted, and decoded uniquely. However, if a channel is particularly noisy and error rates are high, this can be difficult to do while retaining an efficient rate of communication. As a result, alternative methods are used in these situations that help balance the need for good error correction and efficient communication. One such method is *list decoding*, a process that, upon receiving a message, outputs multiple possible decoding options, instead of just one. As a simplified example, consider Microsoft Word's spell checker, which gives multiple possible ways to correct a misspelled word based off of a distance function similar to the Hamming metric:

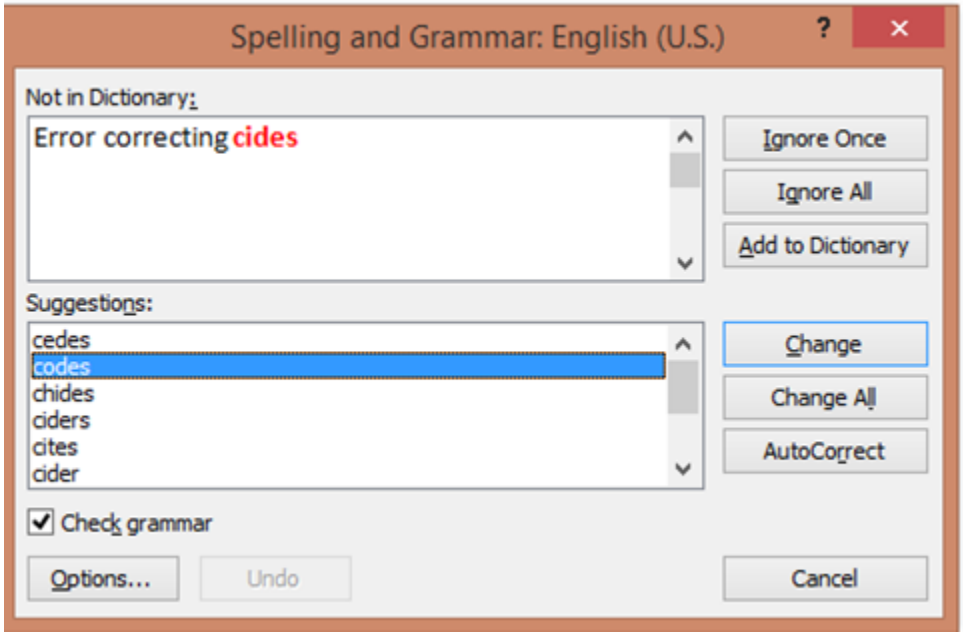

Fig. 1

The hope is that, given some outside context on the receiving end, this list can be narrowed down even further and the original message can be reconstructed up to some standard of

certainty. With this broad goal in mind, we now move to the details of list decoding, starting with the basics provided in Peter Elias's original paper "List Decoding for Noisy Channels," moving to the first nontrivial list decoding algorithm provided by Madhu Sudan, and ending with an application of these results to cryptography.

2 The Basics of List Decoding

2.1 Introduction

In this section we give an overview of the paper "List Decoding for Noisy Channels," written by Peter Elias in 1957. In this paper list decoding is outlined for block codes over the Binary Symmetric Channel (BSC), and several preliminary asymptotic bounds are given. The basics of this process are reminiscent of ordinary block coding and decoding. Given a k element set M of possible messages, we start by creating the *encoding handbook*, which assigns each message in M a unique representation in $\mathbb{F}_2^n$ for some integer $n \geq 1$. As we are transmitting over a noisy channel, we admit the possibility that any error $\mathbf{e}$ from $\mathbb{F}_2^n$ may occur during transmission. As such, we then rank each error vector, associating with each error vector $\mathbf{e}$ an index $r(\mathbf{e})$, $1 \leq r(\mathbf{e}) \leq 2^n$, such that for all $\mathbf{e}_1$, $\mathbf{e}_2$ in $\mathbb{F}_2^n$,

- if $w_H(\mathbf{e}_1) < w_H(\mathbf{e}_2)$, then $r(\mathbf{e}_1) < r(\mathbf{e}_2)$, and
- if $w_H(\mathbf{e}_1) = w_H(\mathbf{e}_2)$, then $r(\mathbf{e}_1) < r(\mathbf{e}_2)$ if and only if the number whose binary representation is $\mathbf{e}_1$ is smaller in magnitude than the number whose binary representation is $\mathbf{e}_2$.

Finally, we create the *decoding codebook*. We first choose an integer l representing the desired length of the outputted decoding list. Given l and a possible received vector $\mathbf{v}$, we then create the list $L_{\mathbf{v}}$ by selecting the l messages whose codewords could have been received as $\mathbf{v}$ by error vectors of the lowest possible rank. I will use the notation that C is a (k, n, l) list decoding scheme if it embeds a messege set M of size k into codewords in $\mathbb{F}_2^n$, and decodes using lists of length l. An example of a $(6, 3, 3)$ list decoding algorithm for the message set $M = \{1, 2, 3, 4, 5, 6\}$ is given below:

Encoding Handbook	
Message	Codeword
1	000
2	100
3	101
4	011
5	110
6	111

Error Rankings	
$\mathbf{e}$	$r(\mathbf{e})$
000	1
001	2
010	3
100	4
011	5
101	6
110	7
111	8

Decoding Handbook	
Received Vector $\mathbf{v}$	Decoding List $L_{\mathbf{v}}$
$\mathbf{v}_1 = 000$	1,2,4
$\mathbf{v}_2 = 100$	1,2,5
$\mathbf{v}_3 = 010$	1,4,5
$\mathbf{v}_4 = 001$	1,3,4
$\mathbf{v}_5 = 110$	2,5,6
$\mathbf{v}_6 = 101$	2,3,6
$\mathbf{v}_7 = 011$	1,4,6
$\mathbf{v}_8 = 111$	4,5,6

Fig. 2: Given the message set $M = \{1, 2, 3, 4, 5, 6\}$, we encode each message as a vector in $\mathbb{F}_2^3$. We then take every possible error vector from $\mathbb{F}_2^3$ and rank them.

To solidify the ranking process, note that $r(010) < r(011)$ because 001 has a lower Hamming weight than 011. Also, $r(010) < r(100)$ because 010 represents the number 2 in binary, which is less than 4, the number represented by 100 in binary. Finally, we have chosen our list size $l = 3$. To create the decoding list for $\mathbf{v}_4$, we examine every way in which a message can be transmitted and $\mathbf{v}_4$ can be received. No message can be sent and received as $\mathbf{v}_4$ with an error of rank 1, but messages 1, 3, and 4 can be sent and received as $\mathbf{v}_4$ with errors of rank 2, 4, and 3 respectively. Thus $L_{\mathbf{v}_4} = 1, 3, 4$.

With the encoding handbook, error ranking, and decoding handbook in hand, the transmission process now becomes simple. Given a message m, we transmit the corresponding codeword $\mathbf{c}$. Some error may occur during transmission, causing the received vector to be $\mathbf{v} = \mathbf{c} + \mathbf{e}$. On the receiving end, the list $L_{\mathbf{v}}$ is created. If m is contained in $L_{\mathbf{v}}$, then we consider the transmission a success. Otherwise the transmission process has failed. With this in mind, we move to figuring out how successful these codes can be.

2.2 Probability of Failure

Let $p < \frac{1}{2}$ be the error probability of a bit transmitted over our BSC, and let $q = 1 - p$. Then given a set of messages M, an encoding handbook, and a decoding handbook, we can determine explicitly the probability of success for the list decoding algorithm. For example, using the code from Figure 2.1.1, and assuming that each message m is transmitted with equal likelihood, we can model each possible situation that our process can go through:

	Error vector	000	001	010	100	011	101	110	111
	Probability	q^3	q^2p	q^2p	q^2p	qp^2	qp^2	qp^2	p^3
Transmitted sequence	000	$\mathbf{v}_1$	$\mathbf{v}_4$	$\mathbf{v}_3$	$\mathbf{v}_2$	$\mathbf{v}_7$	$\mathbf{v}_6$	$\mathbf{v}_5$	$\mathbf{v}_8$
	100	$\mathbf{v}_2$	$\mathbf{v}_6$	$\mathbf{v}_5$	$\mathbf{v}_1$	$\mathbf{v}_8$	$\mathbf{v}_4$	$\mathbf{v}_3$	$\mathbf{v}_7$
	101	$\mathbf{v}_6$	$\mathbf{v}_2$	$\mathbf{v}_8$	$\mathbf{v}_4$	$\mathbf{v}_5$	$\mathbf{v}_1$	$\mathbf{v}_7$	$\mathbf{v}_3$
	011	$\mathbf{v}_7$	$\mathbf{v}_3$	$\mathbf{v}_4$	$\mathbf{v}_8$	$\mathbf{v}_1$	$\mathbf{v}_5$	$\mathbf{v}_6$	$\mathbf{v}_2$
	110	$\mathbf{v}_5$	$\mathbf{v}_8$	$\mathbf{v}_2$	$\mathbf{v}_3$	$\mathbf{v}_6$	$\mathbf{v}_7$	$\mathbf{v}_1$	$\mathbf{v}_4$
	111	$\mathbf{v}_8$	$\mathbf{v}_5$	$\mathbf{v}_6$	$\mathbf{v}_7$	$\mathbf{v}_2$	$\mathbf{v}_3$	$\mathbf{v}_4$	$\mathbf{v}_1$

Fig. 3: The bottom-right 6×8 subtable represents all possible events during transmission. Namely, the cell in row i and column j represents the resulting message caused by the error in column j being introduced during the transmission of the codeword in row i. Such a cell is shaded light grey if this event could be recovered from successfully using the scheme provided in Fig. 2.

From this, we can directly calculate our probability of failure for this specific construct. However, it would be more helpful to give a bound on the error probability for an *optimal* coding scheme with parameters (k, n, l). We define this probability as $P_{opt}(k, n, l)$, which is the minimal attainable error probability of a list decoding scheme with parameters (k, n, l).

As such, consider creating such a table as in Fig. 3 for an arbitrary code. It will have 2^n columns labeled by the 2^n possible error vectors, with e_r, the error in column r,

being the error of rank r. Further, it will have k rows, one for each message. Finally, each row will contain all 2^n vectors in $\mathbb{F}_2^n$. Thus for a given received vector $\mathbf{v}$, exactly l cells labeled $\mathbf{v}$ will be shaded grey, namely the cells in the rows of messages contained in the decoding list for $\mathbf{v}$, $L_{\mathbf{v}}$. So there will be $2^n l$ cells shaded grey in the table, out of a total of $2^n k$ possible cells. Now as the error probability for a single bit is less than $\frac{1}{2}$, a decoding scheme in which the $2^n l$ grey cells appear flush to the left would provided the lowest possible probability of error. For arbitrary parameters (k, n, l), this may not always be acheivable. However, whether it be attainable or not, let $P_t(k, n, l)$ denote the error probability for such a code, which will give a lower bound on $P_{opt}(k, n, l)$.

Proposition 2.1 (Elias, 1957)**.** *Consider a list decoding algorithm with parameters (k, n, l) for which a table as in Fig. 3 is created, and assume that if this table has a grey cell in column r, then all columns i for $i < r$ are colored entirely grey. Let r_0 be the rightmost column which contains a grey cell, i.e.*

$$r_0 k \geq 2^n l > (r_0 - 1)k.$$

Finally, for an integer r, $0 \leq r \leq 2^n$, let $k(r)$ be the unique integer satisfying

$$0 \leq k(r) \leq n \text{ and } \sum_{i=0}^{k(r)} \binom{n}{i} \geq r > \sum_{i=0}^{k(r)-1} \binom{n}{i}.$$

Then $P_t(k, n, l)$, the probability of failure of this scheme, has the following bounds:

$$\sum_{r=r_0}^{2^n} p^{k(r)} q^{n-k(r)} > P_t(k, n, l) \geq \sum_{r=r_0+1}^{2^n} p^{k(r)} q^{n-k(r)}, \tag{1}$$

$$\sum_{i=k(r_0)}^{n} \binom{n}{i} p^i q^{n-i} > P_t(k, n, l) \geq \sum_{i=k(r_0)+1}^{n} \binom{n}{i} p^i q^{n-i}. \tag{2}$$

Proof. Recall that a grey cell corresponds to successful transmission and decoding. Thus the probability of failure is exactly the probability of a transmission ending in an uncolored cell. As there are no grey cells in column r, for $r > r_0$, $P_t(k, n, l)$ is certainly bounded below by the probability of a transmission ending in columns $r_0 + 1, r_0 + 2, \ldots, 2^n$. As each original message being sent is equiprobable, the probability of landing in such a column is merely the probability of that column's corresponding error vector occuring during transmission. Thus

$$P_t(k, n, l) \geq \sum_{r=r_0+1}^{2^n} \text{Prob}(e_r).$$

Note that this bound can be reached if *every* cell in column r_0 is shaded grey. For an upper bound on $P_t(k, n, l)$, we simply act as if no cell in column r_0 was colored grey. This gives the bound

$$\sum_{r=r_0}^{2^n} \text{Prob}(e_r) > P_t(k, n, l) \geq \sum_{r=r_0+1}^{2^n} \text{Prob}(e_r). \tag{3}$$

Now consider $\text{Prob}(e_r)$. The claim is that

$$w_H(e_r) = k(r),$$

so that

$$\text{Prob}(e_r) = p^{k(r)}q^{n-k(r)}.$$

To see this, assume that $w_H(e_r) < k(r)$. Then by definition, there are at least

$$\sum_{i=0}^{k(r)-1} \binom{n}{i}$$

columns preceding column r. Furthermore, there are only

$$\sum_{i=0}^{w_H-1} \binom{n}{i} - 1 < \sum_{i=0}^{k(r)-1} \binom{n}{i} - 1$$

distinct error vectors of weight less than or equal to $w_H(e_r)$ that are not equal to e_r. Thus by the pidgeonhole principle, there must be at least one error vector e_s with $w_H(e_s) > w_H(e_r)$ and $s < r$, which contradicts the error ranking function. Similarly one can rule out $w_H(e_r) > k(r)$, as if this was not the case then there would exist another vector e_s with $w_H(e_s) > k(s)$, which would then lead to another contradiction with e_s. This shows the desired claim. Plugging in the values $\text{Prob}(e_r) = p^{k(r)}q^{n-k(r)}$ into equation (3) now gives the bound in (1). Finally, the bound in (1) gives the bound in (2), as we have both

$$\sum_{r=r_0}^{2^n} p^{k(r)}q^{n-k(r)} \leq \sum_{i=k(r_0)}^{n} \binom{n}{i} p^i q^{n-1}$$

and

$$\sum_{r=r_0+1}^{2^n} p^{k(r)}q^{n-k(r)} \geq \sum_{i=k(r_0)+1}^{n} \binom{n}{i} p^i q^{n-1}.$$

□

Having provided a lower bound on $P_{opt}(k, n, l)$, we now move to an upper bound. To accomplish this, we define $P_{av}(k, n, l)$ to be the *average probability of error* of all list decoding schemes of parameters (k, n, l).

Proposition 2.2 (Elias, 1957)**.** *For parameters (k, n, l), define $Q(r)$ as the average of the probability that the error vector e_r will cause a failure during transmission, where the average is taken over all possible (k, n, l) list decoding schemes. Maintaining the definitions introduced in Proposition 2.1, we have the bound*

$$P_t(k, n, l) \leq P_{av}(k, n, l) \leq \sum_{r=1}^{r_1} p^{k(r)}q^{n-k(r)}Q(r) + P_t(k, n, l).$$

Proof. As $P_t(k,n,l)$ is a lower bound on the optimal error probability for an (k,n,l) code, we certainly have $P_t(k,n,l) \leq P_{av}(k,n,l)$. For the upper bound on $P_{av}(k,n,l)$, note that

$$P_{av}(k,n,l) = \sum_{r=1}^{2^n} p^{k(r)} q^{n-k(r)} Q(r).$$

As $Q(r)$ is an average of probabilities, we have $0 \leq Q(r) \leq 1$. Therefore

$$\begin{aligned} P_{av}(k,n,l) &= \sum_{r=1}^{r_1} p^{k(r)} q^{n-k(r)} Q(r) + \sum_{r=r_1}^{2^n} p^{k(r)} q^{n-k(r)} Q(r) \\ &\leq \sum_{r=1}^{r_1} p^{k(r)} q^{n-k(r)} Q(r) + \sum_{r=r_1}^{2^n} p^{k(r)} q^{n-k(r)}. \end{aligned}$$

Now directly from Proposition 2.1 we have

$$P_t(k,n,l) \leq P_{av}(k,n,l) \leq \sum_{r=1}^{r_1} p^{k(r)} q^{n-k(r)} Q(r) + P_t(k,n,l),$$

as desired. □

We now have

$$P_t(k,n,l) \leq P_{opt}(k,n,l) \leq P_{av}(k,n,l),$$

with further bounds on $P_t(k,n,l)$ and $P_{av}(k,n,l)$. In doing this, we can give asymptotic results on $\frac{P_{av}(k,n,l)}{P_t(k,n,l)}$, which leads to a very interesting conclusion on list decoding. Namely, for large enough n and l, the error probability of almost all (k,n,l) list decoding schemes is near optimal. We reproduce the statement of the theorem here without proof.

Theorem 2.1 (Elias, 1957)**.** *Given k and any $\varepsilon > 0$, it is possible to find n and l large enough so that*

$$1 \leq \frac{P_{av}(k,n,l)}{P_t(k,n,l)} \leq 1 + \varepsilon.$$

With this introduction to list decoding in hand, we now move to a modern result regarding this topic, which comes from the work of Madhu Sudan.

3 List Decoding Reed-Solomon Codes

We now introduce a list decoding algorithm for Reed-Solomon Codes. From here on out, we assume that we are working over an arbitrary finite field $\mathbb{F}$. First, we rephrase the problem as a polynomial reconstruction problem. We will consider a narrow sense Reed-Solomon Code, i.e. one of the form

$$GRS_{n,k+1}(\alpha, \mathbf{1})$$

over a field $\mathbb{F}$ for some $\alpha = (\alpha_1, \ldots, \alpha_n)$ in $\mathbb{F}^n$. Note that a received message will be of the form $\mathbf{r} = (r_1, \ldots, r_n)$, which is the evaluation at α of some polynomial $p(x) \in \mathbb{F}[x]_{k+1}$.

Therefore if we wish to be able to decode $\mathbf{r}$, assuming at most e errors occurred, it suffices to find a polynomial $f(x) \in \mathbb{F}[x]$ of degree at most k which *agrees* with $\mathbf{r}$ in at least $t = n - e$ places. That is to say, there are at least t indices i with

$$f(\alpha_i) = r_i.$$

In the terms of list decoding, we want to be able to find a small list of such polynomials. Thus we have reduced the decoding algorithm to the following problem: Given the parameters n, k, t, and n pairs (α_i, r_i) in $\mathbb{F}^2$, we wish to find all polynomials $f(x)$ in $\mathbb{F}[x]_{k+1}$ such that

$$\Big|\{i : f(\alpha_i) = r_i\}\Big| \geq t.$$

Below we present an algorithm of Sudan which solves the given problem for $t > \sqrt{2nk}$.

The basic algorithm proposed by Sudan can be boiled down to two steps:

1. First, find a bivariate polynomial $Q(x, y) \in \mathbb{F}[x, y]$ of low degree such that

   $$Q(\alpha_i, r_i) = 0$$

   for all i.

2. Factor $Q(x, y)$, and examine all factors of the form $y - f(x)$, for $f(x) \in \mathbb{F}[x]$. We then output all such $f(x)$ with

   $$\Big|\{i : f(\alpha_i) = r_i\}\Big| \geq t.$$

Of course there are many questions to ask regarding this algorithm: First, does such a $Q(x, y)$ exist? Second, for what polynomials $f(x)$ is $y - f(x)$ a factor of $Q(x, y)$? Finally, can we do this efficiently? We answer each question in order.

Proposition 3.1. *For a bivariate polynomial $h(x, y)$ in $\mathbb{F}[x, y]$, let $\deg_x(h)$ and $\deg_y(h)$ denote the highest powers of x and y respectively occurring in $h(x, y)$. Then given n pairs (α_i, r_i) in $\mathbb{F}^2$ and integers d_x, d_y with $(d_x+1)(d_y+1) > n$, there exists a nonzero polynomial $Q(x, y)$ with $d_x = \deg_x(Q)$, $d_y = \deg_y(Q)$, and $Q(\alpha_i, r_i) = 0$ for all i.*

Proof. To see why this is, consider the matrix equation

$$\begin{bmatrix}
\alpha_1^0 r_1^0 & \alpha_1^1 r_1^0 & \dots & \alpha_1^{d_x} r_1^0 & \alpha_1^0 r_1^1 & \dots & \alpha_1^{d_x} r_1^1 & \alpha_1^0 r_1^{d_y} & \dots & \alpha_1^{d_x} r_1^{d_y} \\
\alpha_2^0 r_2^0 & \alpha_2^1 r_2^0 & \dots & \alpha_2^{d_x} r_2^0 & \alpha_2^0 r_2^1 & \dots & \alpha_2^{d_x} r_2^1 & \alpha_2^0 r_2^{d_y} & \dots & \alpha_2^{d_x} r_2^{d_y} \\
 & & & & & \vdots & & & & \\
\alpha_i^0 r_i^0 & \alpha_i^1 r_i^0 & \dots & \alpha_i^{d_x} r_i^0 & \alpha_i^0 r_i^1 & \dots & \alpha_i^{d_x} r_i^1 & \alpha_i^0 r_i^{d_y} & \dots & \alpha_i^{d_x} r_i^{d_y} \\
 & & & & & \vdots & & & & \\
\alpha_n^0 r_n^0 & \alpha_n^1 r_n^0 & \dots & \alpha_n^{d_x} r_n^0 & \alpha_n^0 r_n^1 & \dots & \alpha_n^{d_x} r_n^1 & \alpha_n^0 r_n^{d_y} & \dots & \alpha_n^{d_x} r_n^{d_y}
\end{bmatrix}
\begin{bmatrix}
c_{00} \\ c_{10} \\ \vdots \\ c_{d_x 0} \\ c_{01} \\ \vdots \\ c_{d_x 1} \\ \vdots \\ c_{0 d_y} \\ \vdots \\ c_{d_x d_y}
\end{bmatrix}
=
\begin{bmatrix}
0 \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ 0
\end{bmatrix}$$

where the unknown is the column vector with entries c_{ij}. Now if $(d_x+1)(d_y+1) > n$, then by the Rank-Nullity Theorem, the kernel of the matrix must be nontrivial, implying a nonzero solution to this equation. But now note that we can use the entries in the solution as coefficients in a bivariate polynomial, namely by setting

$$Q(x,y) = \sum_{i}^{d_x} \sum_{j}^{d_y} c_{ij} x^i y^j.$$

From the derivation of these coefficients, it follows that $Q(\alpha_i, r_i) = 0$, as desired. □

Now that we know such a $Q(x,y)$ exists, we want to know which polynomials $f(x)$ have the property that $y - f(x)$ divides $Q(x)$. Specifically, we want to ensure that if $f(\alpha_i) = r_i$ for at least t distinct indices i, then $y - f(x)$ divides $Q(x,y)$.

Proposition 3.2. *Given n pairs (α_i, r_i) in $\mathbb{F}^2$ and a bivariate polynomial $Q(x,y)$ in $\mathbb{F}[x,y]$ with $Q(\alpha_i, r_i) = 0$ for all i, let $f(x)$ be a polynomial in $\mathbb{F}[x]$ such that $f(\alpha_i) = r_i$ for at least t distinct indices i. If $t > \deg_x(Q) + \deg_y(Q)\deg(f)$, then $y - f(x)$ divides $Q(x,y)$.*

Proof. Consider the polynomial $R(x) = Q(x, f(x))$, which is an element of $\mathbb{F}[x]$ whose degree is bounded above by $\deg_x(Q) + \deg_y(Q)\deg(f)$. We know that for at least t distinct values of i, $R(\alpha_i) = Q(\alpha_i, f(\alpha_i)) = Q(\alpha_i, r_i)$, and by choice of Q this gives

$$R(\alpha_i) = Q(\alpha_i, r_i) = 0.$$

Thus $R(x)$ has at least t roots. But since $t > \deg_x(Q) + \deg_y(Q)\deg(f) \geq \deg(R)$, this implies that $R(x)$ is identically zero. Therefore $y = f(x)$ is a root of $Q(x,y)$, implying that $y - f(x)$ divides $Q(x,y)$. □

For the moment we will skip the efficiency of this algorithm, and see how combining the previous two propositions solves our problem. Our list decoding scheme will work with a Reed-Solomon code

$$GRS_{n,k+1}(\alpha, \mathbf{1}),$$

so our list decoding scheme will have parameters (k, n, l) for some list size l. Let us leave l as a variable integer for now. Proposition 3.1 guarantees that we can find a bivariate decoding polynomial $Q(x,y)$ for each transmission, but does not specify what $\deg_x(Q)$ should be or what $\deg_y(Q)$ should be. As we want to find factors of $Q(x,y)$ of the form $y - f(x)$, and as there are at most $\deg_y(Q)$ such factors, let us choose to create $Q(x,y)$ in such a way that $\deg_y(Q) = l$, our list size. By the same proposition, we can take $\deg_x(Q) = \lfloor \frac{n}{l} \rfloor$. Now using Proposition 3.2, we know that our algorithm can find all factors of $Q(x,y)$ of the form $y - f(x)$ so long as our agreement parameter t satisfies

$$t > \deg_x(Q) + \deg_y(Q)\deg(f).$$

With our choice of $\deg_x(Q)$ and $\deg_y(Q)$, and since we can allow $\deg(f)$ to be at most k, we need

$$t > \frac{n}{l} + kl.$$

As we want to minimize the agreement parameter, we need to minimize the right hand side in the above inequality. Since n and k are fixed, we can only modify our list size l. Treating l as a real variable and differentiating, we can see that

$$\frac{d}{dl}\left(\frac{n}{l} + kl\right) = k - \frac{n}{l^2}.$$

This equals zero when

$$l = \pm\left(\frac{n}{k}\right)^{1/2},$$

and as we are looking for a list size we are interested in the positive solution. Finally, as the second derivative is equal to

$$\frac{d}{dl}\left(k - \frac{n}{l^2}\right) = \frac{2n}{l^3},$$

which is positive at $l = \left(\frac{n}{k}\right)^{1/2}$, the function

$$\frac{n}{l} + kl$$

reaches a minimum at $l = \left(\frac{n}{k}\right)^{1/2}$. Plugging in this value to the equation gives

$$\left(\frac{n}{k}\right)^{-1/2} n + k\left(\frac{n}{k}\right)^{1/2} = 2\sqrt{nk}.$$

Therefore the best we can hope for is that $\left(\frac{n}{k}\right)^{1/2}$ is an integer, in which case this can be used as our list size. If this is the case, then we can correctly decode all transmissions in which less than $n - 2\sqrt{nk}$ errors occured. Otherwise, we can choose the nearest integer to $\left(\frac{n}{k}\right)^{1/2}$ as our list size, which will still result in close to the same error handling capabilities.

With this in mind, it remains to show that this process can be done efficiently. Indeed the only difficulty will be finding factors of our bivariate polynomial of the desired form $y - f(x)$, with the degree of $f(x) \leq k$. We give a very brief outline of a polynomial time algorithm that can be used to solve this. Let $Q(x, y)$ be the bivariate polynomial we wish to factor over $\mathbb{F}[x, y]$, and let $E(x)$ be an irreducible polynomial over $\mathbb{F}[x]$ with $\deg(E(x)) > k$. We can then take the extension field

$$K = \mathbb{F}[x]/(E(x)),$$

and take the residue of $Q(x, y)$ mod $E(x)$. This residue, $\overline{Q}(x, y)$, can be thought of as a polynomial in $K[y]$ with coefficients in $\mathbb{F}[x]/(E(x))$. As such, we can use factoring methods for polynomials in one indeterminant to factor $\overline{Q}(x, y)$, such as the Berlekamp algorithm. This will give us a factorization of $\overline{Q}(x, y)$, from which we can determine a factorization of $Q(x, y)$ of the desired form. Moreover, as the Berlekamp algorithm runs in polynomial time, we can do this process fairly quickly.

The above provides a way to effectively list decode Reed-Solomon codes for up to $n - 2\sqrt{nk}$ errors. One should be careful to note that, for small n, this is not always better than the unique decoding of the Reed-Solomon codes, which can deal with up to $\frac{n-k}{2}$

errors. However, this can always be realized as "list decoding," simply by setting our list size to 1 and decoding normally. However, for fixed k, the function of n

$$n - 2\sqrt{nk} - \frac{n-k}{2} = \frac{n - 4\sqrt{nk} - k}{2}$$

is eventually positive for sufficiently large n. So for many n and k, list decoding will give better error handling than unique decoding will, but list decoding will *never* do worse than unique decoding of Reed-Solomon codes.

It also should be stated that several years after Sudan's original algorithm was published, V. Guruswami and Sudan improved upon Sudan's work, allowing the bound on the agreement parameter t to be lowered to $t > \sqrt{nk}$. This allows recovery from up to $n - \sqrt{nk}$ errors using list decoding. The process will not be explained here, but more can be found in [6].

4 An Application

Having gone through an overview of list decoding and the efficient decoding of Reed-Solomon codes using list decoding, we now end with a brief application of list decoding to cryptography. As the cryptographic elements of the application are out of the scope of this paper, we emphasize that this is merely a general discussion of an interesting result due to A. Silverberg, J. Staddon, and J. L. Walker. For a more rigorous treatment of the subject, see their original paper in [3].

The goal of the application is to apply list decoding schemes to protect intellectual property and deter piracy. We say that a collaborating group of individuals are *traitors* if they have rightfully gained access to some piece of intellectual property, but use this access to illegally distribute or "pirate" the software. While this can not always be preemptively defended against, recent advances have made it possible to quickly recognize these actions and even identify some of the traitors using *traitor tracing* algorithms, first developed in the early 1990's. First, given a code C of length n over an alphabet Q with codewords

$$\mathbf{x} = (x_1, \ldots, x_n), x_i \in Q,$$

we say that a subset C_0 of C is a *coalition*. For any coalition C_0, we define the set of *descendants* of C_0, $\text{desc}(C_0)$, as

$$\text{desc}(C_0) = \{\mathbf{w} \in Q^r : w_i = x_i \text{ for some } \mathbf{x} \in C_0, 1 \leq i \leq n\}.$$

For example, if we had a code $C \subset \{1,2,3,4,5\}^5$ which contained the coalition

$$C_0 = \{(1,1,1,1,1), (2,2,2,2,2), (3,3,3,3,3)\}$$

then we would have

$$\text{desc}(C_0) = \{1,2,3\}^5.$$

Note that words in $\text{desc}(C_0)$ are not necessarily in the original code C.

With these definitions, we say that C is a $c-$TA traceability code if, for all coalitions C_0 with $|C_0| \leq c$, if $\mathbf{w} \in \text{desc}(C_0)$, then there is a codeword $\mathbf{x}$ in C_0 such that

$$d_H(\mathbf{x}, \mathbf{w}) < d_H(\mathbf{y}, \mathbf{w})$$

for all $\mathbf{y}$ in $C \setminus C_0$. Intuitively, this says that in a $c-$TA traceability code, any descendant word of a coalition of size at most c must be "closer" to the coalition than the rest of the code in terms of Hamming distance.

Thinking back to what this has to do with pirating software, we can view each codeword as a piece of information given to a legitimate user of some software. Traitors can then pool this information to create a coalition, and try to come up with descendants from their coalition to illegally distribute more information. If the company providing the software could detect a descendant piece of information that was being used, but that was not originally in their code, then they could trace this descendant back to the coalition, and shut down at least some of the traitors based on their results.

With these definitions and goals of traceability, we can relate this to error correction. Namely, we state without proof the following proposition, found in [2]:

Proposition 4.1 (J.N. Staddon, D.R. Stinson, W. Ruizhong, 2002)**.** *Let C be a code of length n, let c be a positive integer, and suppose we have*

$$d_{min}(C) > n - \frac{n}{c^2}.$$

Then

1. *C is a $c-$TA code*
2. *If C_0 is a coalition with $|C_0| < c$ and $\mathbf{w} \in desc(C_0)$, then there exists a codeword in C_0 within distance $n - \frac{n}{c}$ of $\mathbf{w}$. Moreover, every codeword within distance $n - \frac{n}{c}$ of $\mathbf{w}$ is contained in C_0.*

□

With this connection, we can consider a descendant of a coalition to be simply a codeword with some error vector introduced. Thus we can apply results from error correcting codes to turn the traceability problem into one of decoding codewords. Furthermore, while unique decoding of codewords would only give us information on *one* possible traitor in the coalition, list decoding can give us information on *many* possible traitors in the coalition. This suggests that using list decoding algorithms could provide more complete information about the traitors.

To that end, Silverberg, Staddon, and Walker have created an efficient traceability algorithm whose base code is a $c-$TA traceable Reed-Solomon code. Their goal is, given a descendant word $\mathbf{w}$, to find all possible coalitions of size at most c of possible traitors which could have created this word. Loosely put, they start by using the Guruswami-Sudan list decoding algorithm to produce a list of codewords which are close to $\mathbf{w}$ in terms of Hamming distance. Given this set (and an intermediate refinement step), they then look at the tree of coalitions that can be created which contain this base set. In

this tree, they will find all possible coalitions of size at most c from which $\mathbf{w}$ could be a descendant.

In creating this algorithm, Silverberg, Staddon, and Walker were the first to tie list decoding into traceability. They note that with the increasing amount of list decoding algorithms available, such as ones used for concatenated codes or algebraic geometry codes, this area has much room to grow. In particular, they believe that pursuing this further could drastically reduce the run time of traceability algorithms, which would make this protection accessible to many software developers and would help deter the threat of intellectual property theft.

5 Conclusions

Overall, list decoding algorithms seem to be a very promising area of study for error correcting codes. They are very good at increasing error handling capabilities, and thanks to the work of Guruswami, Sudan, and others, these algorithms are becoming more and more efficient with time. Furthermore, with the advent of these faster algorithms, list decoding offers several applications to other fields, such as cryptography. Even in cases in which unique decoding is a necessity, list decoding can still provide thoughtful insight, as we can consider unique decoding as list decoding with list size 1. Due to the advantages provided by this way of solving the error correction problem, it would be beneficial for those in the field to at least become familiar with the topic.

References

[1] J.I. Hall, *Notes on coding theory* [Online], http://www.mth.msu.edu/ jhall/classes/codenotes/Topstuff.pdf (2010).

[2] J.N. Staddon, D.R. Stinson, and R. Wei. *Combinatorial properties of frameproof and traceability codes. IEEE Transactions on Information Theory* **47** (2001), 1042-1049.

[3] A. Silverberg, J. Staddon, and J.L. Walker. *Applications of list decoding to tracing traitors. IEEE Transactions on Information Theory* **49** (2003), 1312-1318.

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

[5] P. Elias. *List decoding for noisy channels. IRE WESCON Convention Record* **2** (1957), 94-104.

[6] V. Guruswami and M. Sudan. *Improved decoding of Reed-Solomon and algebraic geometry codes. IEEE Transactions on Information Theory* **45** (1999), 1757-1767.

[7] V. Guruswami and A.K. Sinop. *Notes 10: List decoding Reed-Solomon codes and concatenated codes* [Online], http://www.cs.cmu.edu/ venkatg/teaching/codingtheory/notes/notes10.pdf (2010).