Error Correcting Codes: Combinatorics, Algorithms and Applications **(Fall 2007)**

Lecture 26: Decoding Concatenated Codes

October 29, 2007

Lecturer: Atri Rudra *Scribe: Michael Pfetsch & Atri Rudra*

In the last lecture, we saw Justesen's strongly explicit asymptotically good code. In today's lecture, we will begin to answer the natural question of whether we can decode concatenated codes up to half their design distance in polynomial time.

1 Decoder for concatenated codes

The concatenation of codes, which was introduced in Lecture 24, is illustrated in Figure 1.

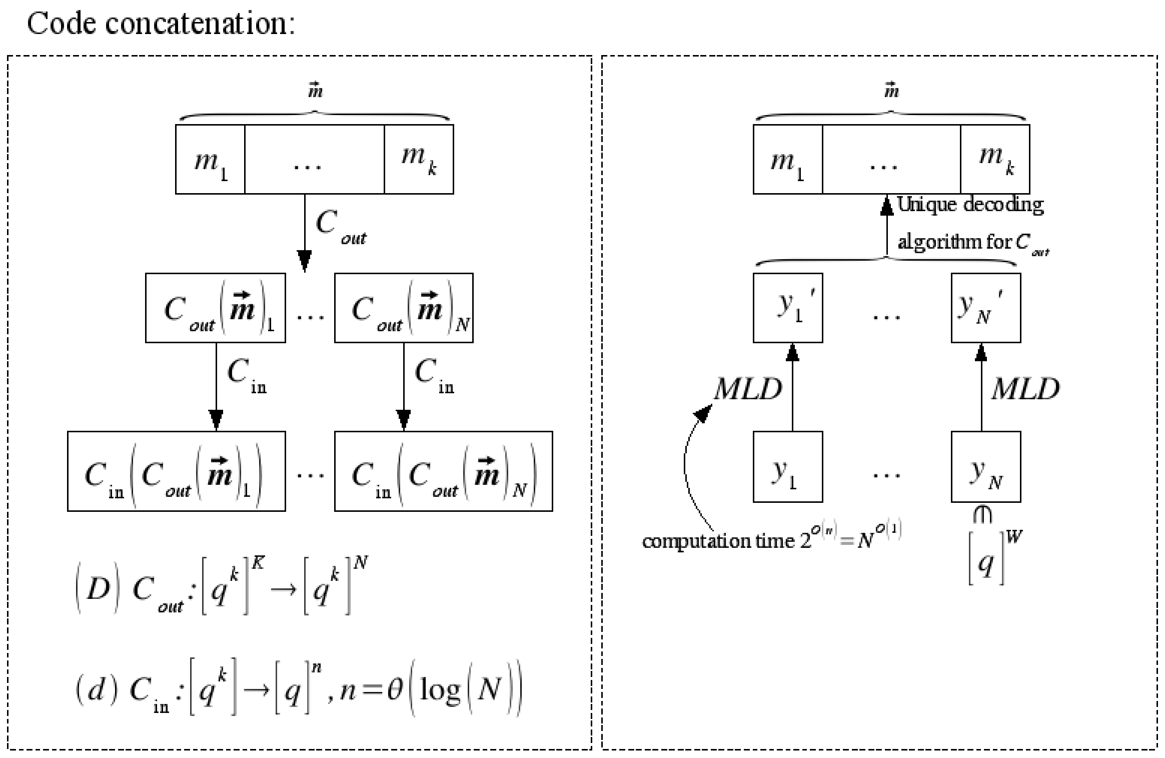
Code concatenation:
$\vec{m}$
m_1 … m_k
C_{out}
$C_{out}(\vec{m})_1$ … $C_{out}(\vec{m})_N$
C_{in}
$C_{in}(C_{out}(\vec{m})_1)$ … $C_{in}(C_{out}(\vec{m})_N)$
$(D)\ C_{out}: [q^k]^K \rightarrow [q^k]^N$
$(d)\ C_{in}: [q^k] \rightarrow [q]^n, n = \theta(\log(N))$
Unique decoding algorithm for C_{out}
y_1' … y_N'
MLD
MLD
y_1 … y_N
$\in [q]^W$
computation time $2^{O(n)} = N^{O(1)}$

Figure 1: Code concatenation and unique decoding of the code $C_{out} \circ C_{in}$.

We begin with a natural decoding algorithm for concatenated codes that "reverses" the encoding process. In particular, the code first decodes the inner code and then decodes the outer code. For the time being let us assume that we have a polynomial time unique decoding algorithm for the outer code. This leaves us with the task of coming up with a polynomial time decoding algorithm for the inner codes. Our task of coming up such a decoder is made easier by the fact that the polynomial running time needs to be polynomial in the *final* block length. This in turn implies that we would be fine if we picked a decoding algorithm that runs in time singly exponential in the inner block length as long as the inner block length is logarithmic in the outer code block length. However, note that the latter is what we have assumed so far and thus, we can use the Maximum Likelihood Decoder (or MLD) for the inner code.

More formally, let the input to the decoder be the vector $\mathbf{y} = (y_1, \cdots, y_N) \in [q^n]^N$. The decoding algorithm is a two step process:

1. Step 1: Compute $\mathbf{y}' = (y_1', \cdots, y_N') \in [q^k]^N$ as follows
$$y_i' = MLD_{C_{in}}(y_i) \quad 1 \le i \le N.$$
2. Step 2: Run the unique decoding algorithm for C_{out} on $\mathbf{y}'$.

It is easy to check that each step of the algorithm above can be implemented in polynomial time. In particular,

1. The time complexity of Step 1 is $O(nq^k)$, which for our choice of $k = O(\log N)$ (and constant rate) for the inner code, is $(nN)^{O(1)}$ time.
2. Step 2 needs polynomial time by our assumption that the unique decoding algorithm for C_{out} takes $N^{O(1)}$ time.

Next, we analyze the performance of the algorithm:

Proposition 1.1. *The decoding algorithm above can correct $< \frac{Dd}{4}$ many errors.*

Proof. We begin the proof by defining a bad event as follows. We say a *bad event* has occurred (at position $1 \le i \le N$) if $MLD_{C_i}(y_i) \ne C_{out}(\mathbf{m})_i$,where $\mathbf{m}$ was the original message and $\Delta(C_{out} \circ C_{in}(\mathbf{m}), \mathbf{y}) < \frac{Dd}{4}$.

Note that if the number of bad events is less than $\frac{D}{2}$, then the decoder in Step 2 will output $\mathbf{m}$. Thus, to complete the proof, we need to figure out when a bad event happens.

Note that if $\Delta(y_i, C_{out}(\mathbf{m})_i) < \frac{d}{2}$ then a bad event *cannot* happen at position i. Since we are trying to upper bound the number of bad events, we will assume that a bad event happens at position i, if
$$\Delta(y_i, C_{out}(\mathbf{m})_i) \ge \frac{d}{2}.$$

Now if the number of bad events is at least $\frac{D}{2}$, then the total number of errors is at least $\frac{D}{2} \cdot \frac{d}{2} = \frac{Dd}{4}$, which is a contradiction. Thus, the number of bad events is strictly less than $\frac{D}{2}$, which completes the proof.

$\square$

Remark 1.2. *The algorithm also works if the inner codes are different, e.g. Justesen codes.*

Thus, we have seen an explicit asymptotically good code that can be encoded and decoded up to some constant fraction of errors in polynomial time. (This of course is modulo the fact that we need a polynomial time unique decoder for the outer code.)

We now state two obvious open questions.

Question 1. *Does there exist a polynomial time unique decoding algorithm for outer codes, e.g. for RS codes?*

Question 2. *Can we correct up to $\frac{Dd}{2}$ errors for the concatenated codes in polynomial time?*

As observed by a student in the lecture, a possible improvement on the decoding algorithm above could be made by taking into account which bits of the code receive more errors. This idea will be useful when we answer Question 2 in the next lecture.

We will now discuss a unique decoding algorithm for Reed-Solomon (RS) codes.

2 Unique decoding for RS codes

In 1960, before polynomial time complexity was regarded as an acceptable notion of efficiency, Peterson designed an $O(N^3)$ time algorithm for the unique decoding of RS codes [1]. This algorithm was the first efficient algorithm for unique decoding of RS codes. The Berlekamp-Massey algorithm, which used shift registers for multiplication, was even more efficient, achieving a computational complexity of $O\left(N^2\right)$. Currently, an even more efficient algorithm, with a computational complexity of $O\left(N \text{poly}(\log N)\right)$, is known. This algorithm is the topic of one of the research survey reports for the course this semester.

The Berlekamp-Welch algorithm, covered under US Patent [2], has a running time complexity of $O\left(N^3\right)$. We will follow a description of the Berlekamp-Welch algorithm provided by Gemmell and Sudan in [3].

Consider the RS code evaluated over $(\alpha_1, \cdots, \alpha_N)$ with dimension K and distance $D = N - K + 1$. Our goal is a describe an algorithm that correct $e < \frac{N-K+1}{2}$ many errors in polynomial time. Let $\mathbf{y} = (y_1, \cdots, y_N) \in \mathbb{F}_q^N$ be the received word. Further, let us assume that there exists a polynomial $P(X)$ such that $\Delta\left(\mathbf{y}, \left(P\left(\alpha_i\right)\right)_{i=1}^N\right) \leq e$. (Note that if such a $P(X)$ exists then it is unique.)

To motivate the algorithm, let us assume that we magically get our hands on a polynomial $E(X)$ such that

$$E\left(\alpha_i\right) = 0 \text{ if and only if } y_i \neq P\left(\alpha_i\right).$$

$E(X)$ is called an *error locator polynomial.* We remark that there exists such a polynomial of degree at most e. In particular, consider the polynomial:

$$E(X) = \prod_{i:y_i \neq P(\alpha_i)} \left(X - \alpha_i\right).$$

Now we claim that for every $1 \leq i \leq N$,

$$y_i E\left(\alpha_i\right) = P\left(\alpha_i\right) E\left(\alpha_i\right) \tag{1}$$

To see why (1) is true, note that if $y_i \neq P(\alpha_i)$, then both sides of (1) are 0 (as $E(\alpha_i) = 0$). On the other hand, if $y_i = P(\alpha_i)$, then (1) is obviously true.

All the discussion above does not seem to have made any progress as both $E(X)$ and $P(X)$ are unknown. Indeed, the task of the decoding algorithm is to find $P(X)$! Next, we perform another seemingly useless task: define $Q(X) \overset{def}{=} P(X) \cdot E(X)$. Note that $Q(X)$ is a polynomial of degree less than or equal to $e + K - 1$. Note that if we can find $Q(X)$ and $E(X)$, then we are done. This is because we can compute $P(X)$ as follows:

$$P(X) = \frac{Q(X)}{E(X)}.$$

The main idea in the Berlekamp-Welch (BW) algorithm is to "forget" what $Q(X)$ and $E(X)$ are meant to be (other than the fact that they are degree bounded polynomials).

2.1 BW decoder

The inputs to the BW decoder are integers $N \geq K \geq 1$, $e < \frac{N-K+1}{2}$, the received word $(y_i, \alpha_i)_{i=1}^{N}$ (such that there exists at most one P (X) of degree at most $k-1$ with $\Delta\left(\mathbf{y}, (P(\alpha_i))_i\right) \leq e$). The output of the BW decoder is either $P(X)$ or a failure. The BW decoder operates in two steps, as follows:

1. Step 1: The BW decoder computes a (non-zero) polynomial $E(X)$ of degree e and a polynomial $Q(X)$ of degree less than or equal to $e + K - 1$ such that the following is true for every $1 \leq i \leq N$:
$$y_i E(\alpha_i) = Q(\alpha_i).$$
If such polynomials do not exist, then the output will be a failure.

2. Step 2: If $E(X)$ does not divide $Q(X)$ then the output will be a failure. Otherwise, define $P'(X) = \frac{Q(X)}{E(X)}$. If $\Delta(\mathbf{y}, (P'(\alpha_i))_i) \leq e$, then output $P'(X)$ else output failure.

In the next lecture, we will focus on the following two questions regarding the BW decoder:

1. Is the algorithm correct?

2. What is the time complexity of the algorithm?

References

[1] W. W. Peterson. ”Encoding and Error-Correction Procedures for the Bose-Chaudhuri Codes.” *IRE Transactions on Information Theory*, vol. 6, pp. 459–470, Sept. 1960.

[2] L. Welch and E. R. Berlekamp. “Error correction of algebraic block codes,” U.S. Patent 4 633 470, Dec. 1986.

[3] P. Gemmell and M. Sudan. “Highly resilient correctors for multivariate polynomials,” *Inform. Processing Lett.*, vol. 43, no. 4, pp. 169–174, 1992.