6.897 Algorithmic Introduction to Coding Theory November 27, 2002

Lecture 21

Lecturer: Madhu Sudan *Scribe: Deniss Čebikins*

1 Applications of error-correcting codes

Since the main purpose of computers is to manipulate information, it is clear that every computer must have some information preserving capabilities. The original motivation for studying error-correcting was to develop methods of protecting data from occasional transmission errors. Recent progress in coding theory revealed quite fascinating applications of these codes to other areas in computer science and mathematics. In this lecture, we take a look at how certain combinatorial properties of error-correcting codes can be used in solving seemingly unrelated problems.

2 Communication complexity

Suppose that Alice has a string $x \in \{0,1\}^k$ and Bob has a string $y \in \{0,1\}^k$. They want to find the value of $f(x,y)$ for some boolean function $f : \{0,1\}^k \times \{0,1\}^k \to \{0,1\}$ which is known to both of them. For example, if they want to determine whether their strings x and y are equal, then f is the function EQ defined by

$$EQ(x,y) = \begin{cases} 1, & \text{if } x = y, \\ 0, & \text{if } x \neq y. \end{cases}$$

Alice and Bob also want to minimize the number of bits they need to send to each other in order to compute $f(x,y)$. For now, assume that the interaction occurs according to a certain deterministic procedure, which we call a *protocol*, in which each message that Alice (resp. Bob) transmits depends only on x (resp. y) and the previous messages. More formally, a protocol can be described as follows:

- First, Alice computes $a_1 \leftarrow A_1(x)$ and sends it to Bob;
- Bob computes $b_1 \leftarrow B_1(y, a_1)$ and sends it back to Alice;
- Alice computes $a_2 \leftarrow A_2(x, a_1, b_1)$ and sends it to Bob;
- Bob computes $b_2 \leftarrow B_2(y, a_1, b_1, a_2)$ and sends it to Alice;
- Etc.

Here A_1, A_2, ..., and B_1, B_2, ..., are sequences of deterministic algorithms modeling Alice's and Bob's behavior, respectively. We require that the lengths $|a_i|$ and $|b_i|$ of messages transmitted at each step depend only on the protocol P. We also require that once a person determines $f(x,y)$, he or she sends it to the other person at the last step of the protocol.

For a fixed k, let the *size* of a protocol P be the maximum total number of bits, over all possible pairs $x, y \in \{0,1\}^k$, that Alice and Bob transmit during the execution of the

protocol. We define the *communication complexity* of a boolean fuction f, denoted $\mathrm{CC}(f)$, to be the smallest size of a protocol at the end of which both parties know $f(x, y)$.

An easy way to find $f(x, y)$ in this scenario is to have Alice send to Bob the entire string x, and then have Bob compute $f(x, y)$ and send the obtained value back to Alice. In this protocol, a total of $k + 1$ bits is transmitted. Thus, we get a bound

$$\mathrm{CC}(f) \leq k + 1$$

valid for all boolean functions f.

It is natural to ask if the above bound is tight. That is, is there a function f whose communication complexity is exactly $k + 1$? The answer is "yes"; moreover, the equality function $EQ(x, y)$ has this property.

Proposition 1 *$CC(EQ) = k + 1$.*

Proof Suppose some protocol $P = (A_1, B_1, A_2, B_2, \ldots)$ of size at most k determines $EQ(x, y)$. Then the interaction history $(a_1, b_1, a_2, b_2, \ldots)$ before one of the parties determines $EQ(x, y)$ can take at most 2^{k-1} values. Since there are 2^k strings in $\{0, 1\}^k$, there exist distinct strings $x, x' \in \{0, 1\}^k$ such that running P on inputs (x, x) and (x', x') produces the same interaction history $(a_1, b_1, \ldots)$. Consider the interaction history $(a'_1, b'_1, \ldots)$ resulting from running P on input (x, x'). We get

$$a'_1 = A_1(x) = a_1$$

$$b'_1 = B_1(x', a'_1) = B_1(x', a_1) = b_1$$

$$a'_2 = A_2(x, a'_1, b'_1) = A_2(x, a_1, b_1) = a_2$$

$$\ldots$$

Thus running P on (x, x') produces the same interaction history as running P on (x, x), so we must have $EQ(x, x') = EQ(x, x)$, which is a contradiction. We conclude that $\mathrm{CC}(EQ) = k + 1$. ■

3 Probabilistic communication complexity

Let us modify the scenario of the previous section by allowing Alice and Bob to generate random bits in private. That is, the parties are permitted to use random bits in producing messages, but they are not allowed to share these random bits with the other party.

This modification is useful only if we make our requirement of Alice and Bob determining $f(x, y)$ somewhat less restrictive. For this reason, let us accept the following in this scenario: a probabilistic protocol described above determines the function f if for any $x, y \in \{0, 1\}^k$, the probability that the parties compute $f(x, y)$ at the last step correctly is at least $2/3$. We define the *probabilistic communication complexity* of a boolean function f, denoted $\mathrm{PCC}(f)$, to be the size of a smallest probabilistic protocol that determines f.

It turns out that probabilistic communication complexity of a boolean function can be significantly smaller than its "regular" communication complexity. Again, we consider the example of the equality function $EQ(x, y)$.

Consider the following probabilistic protocol:

- Alice chooses an index $i \in \{1, 2, \ldots, k\}$ at random and sends (i, x_i) to Bob (here x_i denotes the i-th bit of x);

- Bob sends 1 to Alice if $x_i = y_i$, and 0 otherwise.

Clearly, if $x = y$, then Bob always sends the correct value of $EQ(x, y)$ to Alice at the last step. If $x \neq y$, then Bob sends the correct value of $EQ(x, y)$ if x and y differ in the i-th bit, which is true with probability of at least $1/k$.

This is not quite meeting our goal, but we can use error-correcting codes to enhance the above construction and design small size probabilistic protocols determining EQ.

Proposition 2 *$PCC(EQ) = O(\log k)$.*

Proof Let $E : \{0,1\}^k \to \mathbb{F}_q^{3k}$ be the encoding function of a Reed-Solomon $[3k, k, 2k]_q$-code, where q is slightly greater than $3k$. Consider the following probabilistic protocol:

- Alice chooses an index $i \in \{1, 2, \ldots, 3k\}$ at random and sends $(i, E(x)_i)$ to Bob, where $E(x)_i$ denotes the i-th coordinate of $E(x)$ (we regard $\mathbb{F}_q^{3k}$ as a $3k$-dimensional vector space over $\mathbb{F}_q$);

- Bob sends 1 to Alice if $E(x)_i = E(y)_i$, and 0 otherwise.

If $x = y$, then Bob always sends the correct value of $EQ(x, y)$ to Alice at the last step. If $x \neq y$, then $E(x)$ and $E(y)$ differ in at least $2k$ out of $3k$ coordinates, so the probability that $E(x)$ and $E(y)$ differ in the i-th coordinate and hence Bob sends the correct value of $EQ(x, y)$ to Alice is at least $2/3$, as desired.

It remains to compute the number of bits transmitted in the above protocol. We need $\log 3k$ bits to transmit i, and since $E(x)_i \in \mathbb{F}_q$, we need $\log q \approx \log 3k$ bits to transmit $E(x)_i$. Finally, Bob transmits one more bit at the last step, so

$$\mathrm{PCC}(EQ) \leq \log 3k + \log q + 1 = O(\log k).$$

■

We conclude the section by considering the scenario in which Alice and Bob share their random bits. In this case, Alice and Bob are first given the function f. Then, they generate a random string s together. Afterwards, Alice is given x and Bob is given y, and they determine $f(x, y)$ according to a deterministic protocol in which they are allowed to use their knowledge of s.

It is clear that in this scenario, Alice and Bob can determine $EQ(x, y)$ by transmitting $O(\log k)$ bits using the protocol of the above proposition; the only difference is that the index i is generated by the two parties together rather than by Alice alone. We can also increase the efficiency of the protocol since the index i does not need to be transmitted. If we choose a binary code with relative minimum distance of at least $2/3$, then Alice and Bob need to transmit only two bits, namely, $E(x)_i$ and the bit sent by Bob at the last step.

4 Secret sharing

In this section, we discuss an application of error-correcting codes to the problem of secret sharing. Suppose we have a piece s of secret information, and we want to "spread" this information across n parties, $p_1, \ldots, p_n$. We do so by generating a random string r and letting the party p_i know the value of $s_i = f_i(s, r)$, where f_i is a function which is up to us to define. We want our secret sharing scheme to satisfy two basic properties:

- Secrecy: no subset $\{p_{i_1}, \ldots, p_{i_t}\}$ of t parties can determine s from their shares $s_{i_1}, \ldots, s_{i_t}$ of the secret;

- Recovery: every subset $\{p_{i_1}, \ldots, p_{i_T}\}$ of T parties can determine s from their shares $s_{i_1}, \ldots, s_{i_T}$ of the secret.

Here $1 \leq t < T \leq n$ are fixed parameters. The following construction due to Shamir achieves the desired properties in the "ideal" case $T = t + 1$. The secret information s in this construction is an element of the field $\mathbb{F}_q$.

- Given s, choose non-zero elements $\alpha_1, \ldots, \alpha_t$ of $\mathbb{F}_q$ and distinct elements $\beta_1, \ldots, \beta_n$ of $\mathbb{F}_q$;

- Set $p(x) = \alpha_t x^t + \ldots + \alpha_1 x + s$ and $s_i = (\beta_i, p(\beta_i))$.

We omit the proof of the validitiy of this scheme. Instead, we consider another scheme, which is also constructed using an error-correcting code.

Let C be a linear $[n + 1, k, d]_q$-code. Consider the following secret sharing scheme (s is a member of the alphabet of C):

- Choose a codeword $(c_0, c_1, \ldots, c_n) \in C$ such that $c_o = s$;

- Set $s_i = c_i$.

Any T parties can recover s from their shares of the secret if it is possible to decode a codeword with $n+1-T$ erasures (one erasure at c_0, and $n-T$ more erasures corresponding the other $n - T$ parties). Such decoding is possible if $n - T + 1 < d$.

It can also be shown that if the code $C^\perp$ dual to C has minimum distance Δ, then no subset of $\Delta - 2$ parties can determine s from their shares of the secret.

Thus, if $n - T + 1 \leq d - 1$ and $t \leq \Delta - 2$, then the above secret sharing scheme has the desired properties. To show that the two conditions can be satisfied in the "ideal" case $T = t + 1$, let C be a Reed-Solomon $[n + 1, t + 1, n - t + 1]_q$-code. Then its dual is an $[n + 1, n - t, t + 2]$-code. Thus, $\Delta = t + 2$ and $d = n - t + 1 = n - T + 2$, so the above conditions are satisfied.