EECS 121: Coding for Digital Communication & Beyond **Fall 2013**

Lecture 24 — November 25

Lecturer: K. V. Rashmi *Scribe: Preetum Nakkiran*

24.1 Overview

In this lecture, we will investigate the piecewise-linearity of the storage/bandwidth tradeoff curve [1], discuss an explicit repair-by-transfer MBR code [3], and finally describe the explicit product-matrix construction of Rashmi, Shah, Kumar [2] for general MBR codes.

24.2 Storage/Bandwidth Tradeoff

Recall from last lecture, we derived a fundamental tradeoff between storage space and repair-bandwidth for a fixed file size, based on the min-cut bound from network coding:

$$B \leq \sum_{i=0}^{k-1} \min(\alpha, (d-i)\beta)$$

α: Number of symbols stored by each node.
β: Number of symbols transferred from each helper-node to a failed node.
B: Total file size (number of symbols).
k: Connecting to any k nodes and downloading α from each is sufficient to reconstruct the full data.
d: Connecting to any d nodes and downloading β from each is sufficient to repair a failed node.

Further, noticing that optimal codes had α between $(d-(k-1))\beta$ (for MSR) and $d\beta$ (for MBR), we considered parametrizing points on the tradeoff curve by $p \in \mathbb{Z}$ and $\theta \in \mathbb{R}$, where:
$\alpha = (d-p)\beta - \theta$
$p = \{0 \dots (k-1)\}$
$\theta \in [0, \beta)$
Using this parametrization, we have seen that the min-cut bound is equivalent to (some f):

$$B \leq (p+1)\alpha + f(p)\beta$$

Claim: For two points (α_1, β_1) and (α_2, β_2) on the optimal tradeoff curve and with the same $p = p^*$, all points on the line between them are also optimal, and have the same $p = p^*$.

Proof: Let $\lambda \in [0,1]$ and $\bar{\lambda} = 1 - \lambda$. Points on the line have the form:
$(\alpha', \beta') = (\lambda\alpha_1 + \bar{\lambda}\alpha_2, \lambda\beta_1 + \bar{\lambda}\beta_2)$.

Since both points have the same $p = p^*$, we can write:
$\alpha_1 = (d - p^*)\beta_1 - \theta_1$ and $\alpha_2 = (d - p^*)\beta_2 - \theta_2$.
Notice that $\alpha' = \lambda\alpha_1 + \bar{\lambda}\alpha_2 = (d - p^*)(\lambda\beta_1 + \bar{\lambda}\beta_2) - (\lambda\theta_1 + \bar{\lambda}\theta_2) = (d - p^*)\beta' - \theta'$.
For $\theta' = \lambda\theta_1 + \bar{\lambda}\theta_2 \in [0, \beta')$.

Therefore, the point (α', β') has the same $p = p^*$ as the original points.

Further, since both original points were optimal, they satisfied the min-cut bound with equality, and we can write:
$B = (p^* + 1)\alpha_1 + f(p^*)\beta_1$
$B = (p^* + 1)\alpha_2 + f(p^*)\beta_2$
Manipulating:

$$\begin{aligned} \lambda B + \bar{\lambda} B &= \lambda[(p^* + 1)\alpha_1 + f(p^*)\beta_1] + \bar{\lambda}[(p^* + 1)\alpha_2 + f(p^*)\beta_2] \\ &= (p^* + 1)(\lambda\alpha_1 + \bar{\lambda}\alpha_2) + f(p^*)(\lambda\beta_1 + \bar{\lambda}\beta_2) \\ B &= (p^* + 1)\alpha' + f(p^*)\beta' \end{aligned}$$

Therefore, the point (α', β') is also optimal, since it satisfies the parameterized min-cut bound $(B \leq (p+1)\alpha + f(p)\beta)$ with equality.

This demonstrates the **piecewise-linearity** of the tradeoff curve: Every value of p (from 0 to $k-1$) corresponds to a linear segment of the tradeoff curve (with endpoints at $\theta = 0$).

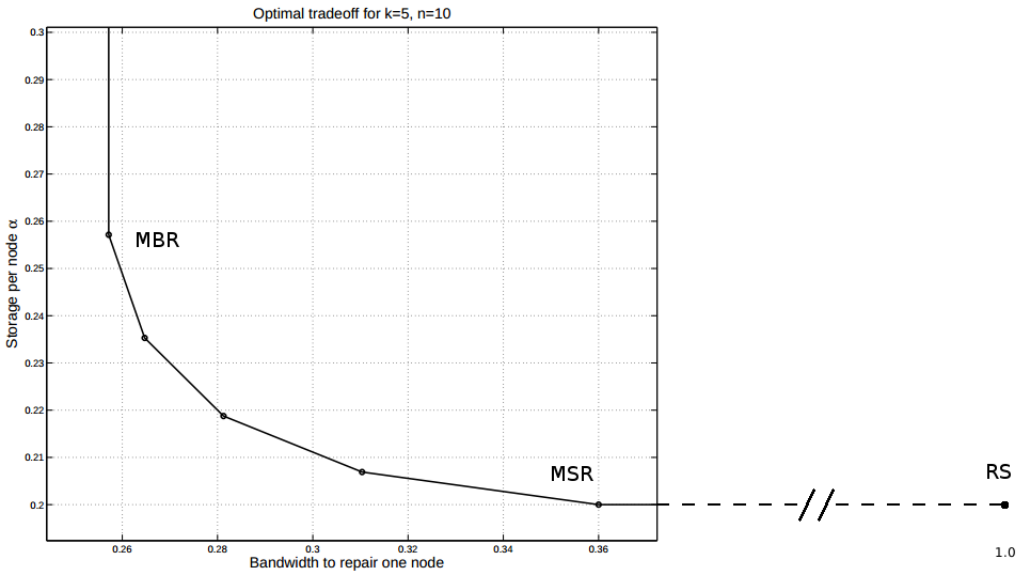

Figure 24.1: Piecewise-linear tradeoff between per-node storage (α) and repair-bandwidth ($d\beta$) for regenerating codes. Normalized to file-size $B = 1.0$. The Reed-Solomon operating point (RS) is included. Adapted from [1]

Codes which lie on the optimal tradeoff curve are called "regenerating codes". At the two endpoints of the curve are MSR codes (Minimum-Storage-Regenerating codes) where α is minimum, and MBR codes (Minimum-Bandwidth-Regenerating codes) where $d\beta$ is minimum. Reed-Solomon codes have the same storage as MSR codes, but with much greater repair-bandwidth (namely, the entire file).

Notice that at the MSR point we have $\alpha = \frac{B}{k}$, so all the k nodes participating in data recovery store only what is necessary ($\frac{1}{k}$ of the total information). And at the MBR point, $\alpha = d\beta$, so all d nodes participating in node-repair send only what is necessary ($\frac{1}{d}$ of the final stored information α).

The intuition is that for lower α, the amount of overlapping information stored in nodes is less, so repair requires downloading more data (nodes effectively send the intersection of their subspace with the failed-node's subspace, in a manner that will be clear from the Product-Matrix framework below).

24.3 Explicit Regenerating Codes

24.3.1 MBR Repair-by-transfer

We will first consider an example of an explicit MBR code, with the property that node repair involves no computation. Consider the case: $n = 5, k = 3, d = 4, \beta = 1, \alpha = d\beta = 4, B = kd - \binom{k}{2} = 9$.

Working in binary, take the 9 message bits: $c_1 \dots c_9$ and generate a 10th parity symbol $c_{10} = c_1 \oplus c_2 \oplus \dots c_9$. Notice that this "outer-code" has the 9/10 property: any 9 of 10 symbols can be used to reconstruct the entire message.

Now visualize the next coding stage as a 5-clique: 5 vertices representing nodes, and $\binom{5}{2} = 10$ total edges between them. Now assign each symbol (including the parity) to an edge. Each node will store all the symbols from their incident edges: 4 symbols each.

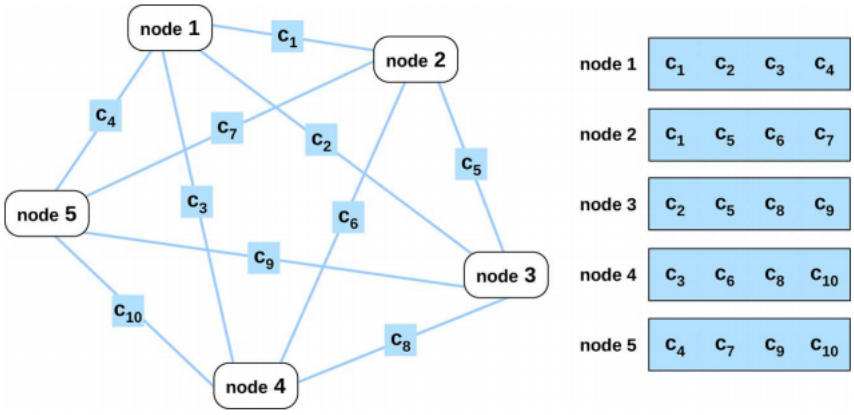

Figure 24.2: The repair-by-transfer code visualized as a clique. From [3]

Let us verify the regenerating-code properties:

1. Data Collection: Connecting to any $k = 3$ nodes should be sufficient to recover all 9 message symbols. Notice that any 3 nodes contain a total of $\alpha k = dk = 12$ total symbols, but 3 of these symbols are repeated, since there are $\binom{k}{2} = 3$ edges between any set of 3 nodes. So there are only $dk - \binom{k}{2} = 12 - 3 = 9$ unique symbols. But by the 9/10 property, we know that any 9 of the 10 coded symbols can be used to recover the entire message.

2. Node Repair: When a node fails, it must recover its $\alpha = 4$ symbols by connecting to any $d = 4$ nodes. In this case, it $d = n - 1$, so it must connect to all other nodes, and recover 4 symbols corresponding to its incident edges. Since every incident edge also connects to one other node, each of the surviving nodes can simple send "along the edge" the corresponding symbol, and the repaired node simply stores the 4 received symbols – no computation required.

So this is a valid MBR code.

Note that we have not restricted the code by fixing $\beta = 1$ – if we have an optimal code for $\beta = 1$, we can construct optimal codes for higher β' simply by using the original code β' times (for example, for $\beta' = 2$, split the file into 2 pieces, and encode each piece with the optimal $\beta = 1$ code, then store both codes on each node).

Further, the example above admits a generalization to MBR codes for any parameters $(n, k, d = n - 1)$. The only modification is in the outer-code, where instead of XORing to get a 9-out-of-10 property, we need to use a Reed-Solomon-like code to get a B-out-of-$\binom{n}{2}$ property (any B symbols from $\binom{n}{2}$ coded symbols can be used to recover B uncoded symbols). To complete the generalization:
Place one coded symbol on each of the $\binom{n}{2}$ edges in the n-clique. Each node will have $n - 1 = d = \alpha$ incident edges, and store these symbols.
Data-Collection: Any k nodes will store between them $dk - \binom{k}{2}$ unique symbols. But $B = dk - \binom{k}{2}$, so we can recover our message from these coded symbols (using our RS-like code).
Repair: $d = n - 1$, so all the surviving nodes send their symbols "along the edges", just as before.

Exercise for the reader: Work out an example of data-collection for a $[n, k, d] = [5, 2, 3]$ repair-by-transfer code. In this case, $B = 5$, and you will need to use a $(10, 5)$ Reed-Solomon code.

24.3.2 Product-Matrix Codes

Note: This section is heavily based on the Product-Matrix paper by Rashmi, Shah, Kumar [2].

Product-Matrix codes are new constructions of linear regenerating codes capable of achieving both MBR and MSR points. We will first develop the general structure of product-matrix codes, then describe the explicit construction achieving the MBR point. All product-matrix codes operate with $\beta = 1$ – as mentioned above, this is sufficient to construct optimal codes for higher β.

In the product-matrix framework, codewords can be represented by the $(n \times \alpha)$ code matrix C:

$$\underset{n \times \alpha}{C} = \underset{n \times d}{\Psi} \quad \underset{d \times \alpha}{M}$$

The encoding-matrix Ψ is fixed beforehand, and the message matrix M is filled with B distinct message symbols in a redundant manner. Node i stores α symbols: the i^{th} row of Ψ times M:

$$c_i^t = \psi_i^t M$$

Here, the superscript t denotes the transpose operation.

In data-collection, every participating node sends the α symbols it stores: $\psi_j^t M$. So the data-collector receives the product of k rows of Ψ times M:

$$\Psi_{DC} M$$

Where Ψ_{DC} is the submatrix of Ψ containing the k rows corresponding to connected nodes. From this product, the data-collector performs linear operations to recover the B unique symbols in M.

For node repair, the failed node f connects to d other helper-nodes, each of which send $\beta = 1$ symbol: the inner product of a particular vector u_f $(\alpha \times 1)$ with the helper-node's stored symbols:

$$c_i^t u_f = \psi_i^t M u_f$$

The failed node thus receives the product matrix:

$$\Psi_{\text{repair}} M u_f$$

Where Ψ_{repair} is a submatrix of Ψ containing d rows corresponding to the helper nodes. The failed node then performs linear operations to recover its stored symbols $c_f^t = \psi_f^t M$.

24.3.3 MBR Product-Matrix Construction

This construction covers all the parameters $[n, k, d]$ at the MBR point. Recall that for any optimal $[n, k, d, \beta = 1]$ regenerating code at the MBR point, we must have (by the min-cut

bound):

$$B = \binom{k+1}{2} + k(d-k)$$

$$\alpha = d$$

Construct the message matrix M as follows. First populate the upper-triangular half of a $(k \times k)$ matrix S with $\binom{k+1}{2}$ distinct message symbols, and the strictly lower-triangular half with duplicate symbols to make S symmetric. Then populate a $(k \times (d-k))$ matrix T with $k(d-k)$ other distinct message symbols. Construct M as the symmetric matrix:

$$M = \begin{bmatrix} S & T \\ T^t & 0 \end{bmatrix}$$

Notice that by construction of S and T, the message matrix contains exactly $\binom{k+1}{2}+k(d-k) = B$ unique symbols.

The encoding matrix is constructed as:

$$\Psi = \begin{bmatrix} \Phi & \Delta \end{bmatrix}$$

For $(n \times k)$ matrix Φ and $(n \times (d-k))$ matrix Δ with the following two properties:

1. Any d rows of Ψ are linearly independent.
2. Any k rows of Φ are linearly independent.

This can be achieved, for example, by a Vandermonde matrix [1] :

$$\Psi = \begin{bmatrix} 1 & \omega_1 & \omega_1^2 & \cdots & \omega_1^{d-1} \\ 1 & \omega_2 & \omega_2^2 & \cdots & \omega_2^{d-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega_n & \omega_n^2 & \cdots & \omega_n^{d-1} \end{bmatrix}$$

Where $\omega_1, \cdots, \omega_n$ are distinct and the field size is at least n.

Node-Repair (for failed node f):
In this case, use $u_f = \psi_f$. Every helper node i computes the product: $\psi_i^t M \psi_f$ The failed node thus receives the product $\Psi_{\text{repair}} M \psi_f$. But by property (1) of Ψ, we know Ψ_{repair} (some d rows of Ψ) is invertable, so the failed node can invert to find $M\psi_f$. Now, since M is

[1] Recall that a Vandermonde matrix is essentially a matrix that evaluates polynomials, and can be used (for example) as the generator matrix of a Reed-Solomon code. If M is $n \times d$ Vandermonde, and x is a vector of polynomial coefficients, then Mx is the vector of evaulating the degree $d-1$ polynomial specified by x at the points $w_1, w_2 \ldots w_n$. That is, $[Mx]_i = x_0 + x_1(w_i) + x_2(w_i^2) + x_3(w_i)^3 \ldots$. Notice that any d rows of M are necessarily linearly-independent, since a degree $d-1$ polynomial is uniquely specified by d evaluations.

symmetric, it can transpose to find $(M\psi_f)^t = \psi_f^T M^t = \psi_f^t M$. But this is exactly the failed node's stored data! So node-repair is successful.

Data Collection

Let

$$\Psi_{DC} = \begin{bmatrix}\Phi_{DC} & \Delta_{DC}\end{bmatrix}$$

be the $(k \times d)$ submatrix of k rows of Ψ corresponding to the connected nodes. The data-collector receives the product:

$$\Psi_{DC} M = \begin{bmatrix}\Phi_{DC} & \Delta_{DC}\end{bmatrix} \begin{bmatrix} S & T \\ T^t & 0 \end{bmatrix} = \begin{bmatrix}\Phi_{DC} S + \Delta_{DC} T^t & \Phi_{DC} T\end{bmatrix}$$

Since Φ_{DC} is invertable by property (2), we can first recover T from the right block, then subtract out T's influence and recover S from the left block. So data-collection is successful!

Therefore this product-matrix construction yields an optimal linear regenerating code at the MBR point.

Bibliography

[1] Alexandros G. Dimakis, P. Brighten Godfrey, Yunnan Wu, Martin J. Wainwright, and Kannan Ramchandran. Network coding for distributed storage systems. In *IEEE Transactions on Information Theory, vol. 56, no. 9*, September 2010.

[2] K. V. Rashmi, Nihar B. Shah, and P. Vijay Kumar. Optimal exact-regenerating codes for distributed storage at the msr and mbr points via a product-matrix construction. *IEEE Transactions on Information Theory*, 2011.

[3] Nihar B. Shah, K. V. Rashmi, P. Vijay Kumar, and Kannan Ramchandran. Distributed storage codes with repair-by-transfer and non-achievability of interior points on the storage-bandwidth tradeoff. *IEEE Transactions on Information Theory*, 2012.