Computation Codes - A New Tool for Cooperative Communication

Bobak Nazer

(Joint work with Michael Gastpar)

Wireless Foundations
Department of Electrical Engineering and Computer Sciences
University of California, Berkeley

May 7, 2008
Overview

Main ideas for this talk:

- New coding technique for efficient and reliable distributed computation over the wireless channel
Overview

Main ideas for this talk:

- New coding technique for efficient and reliable distributed computation over the wireless channel

- New communication strategy that allows relays in a network to reliably “compute-and-forward” for higher throughputs.
Point-to-Point Communication

- We know the capacity for an AWGN channel:

\[ C = \frac{1}{2} \log (1 + \text{SNR}) \]

- To prove this, we can just use a random codebook.
- Insight: use codes with good minimum distance.
Multiple-Access Communication

- We also know the **capacity** for a Gaussian multiple-access channel. The symmetric capacity per user is:

\[
C_{\text{USER}} = \frac{1}{4} \log (1 + 2\text{SNR})
\]

- Again, we can show this by using **random codebooks**.
- Insight: orthogonalize users.
General Wireless Networks

- If we simply apply these ideas to wireless networks, we can get a communication strategy.
- Roughly speaking, this tells us to:
  - Establish reliable links to nearby users.
  - Treat other transmissions as unwanted interference.
  - Basically, make wireless networks look like wired ones.
Need for Cooperation

- In many cases, we can do much better with cooperative communication.

- Examples: distributed MIMO, distributed beamforming, cooperative diversity, network coding.

- Many of these schemes implicitly use the additive structure of the wireless channel to appropriately combine signals.

- Computation codes are designed to exploit the addition provided by the channel.
Motivating Example: Mod-2 Adder

- Binary inputs $S_1$ and $S_2$. Receiver only wants parity $S_1 \oplus S_2$
- Channel takes mod-2 sum of its (binary) inputs.
- Standard multiple-access: requires sending $S_1$ and $S_2$ separately, rate $= \frac{1}{2}$
- Uncoded transmission: channel computes for us, rate $= 1$
Now let’s bring in noise...

- Uncoded transmission: we can only get a noisy sum $S_1 \oplus S_2 \oplus Z$
- Standard multiple-access: noise-free sum, encoders compete for the channel
- Computation coding uses the channel to add while protecting against noise.
Computation Coding

- **Key idea**: Use the same linear code at each encoder
- Write this linear code as a generator matrix $G$

<table>
<thead>
<tr>
<th>Codewords</th>
<th>Channel Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1 = s_1 G$</td>
<td>$y = s_1 G \oplus s_2 G \oplus z$</td>
</tr>
<tr>
<td>$x_2 = s_2 G$</td>
<td>$= (s_1 \oplus s_2)G \oplus z$</td>
</tr>
<tr>
<td>$x_{SUM} = (s_1 \oplus s_2)G$</td>
<td>$= x_{SUM} \oplus z$</td>
</tr>
</tbody>
</table>

- Channel adds up the codewords and receiver only sees the codeword for the sum!
- *(Optimal for our little example.)*
Decoding Functions of Codewords
Decoding Functions of Codewords
Decoding Functions of Codewords
Decoding Functions of Codewords

- Sum of codewords is not a codeword.
- Must decode individual codewords.
Decoding Functions of Codewords

- Sum of codewords is not a codeword.
- Must decode individual codewords.
Decoding Functions of Codewords

- Sum of codewords is **not** a codeword.
- Must decode individual codewords.
Decoding Functions of Codewords

- Sum of codewords is **not** a codeword.
- Must decode individual codewords.

- Sum of codewords is a codeword.
- Can decode linear functions of codewords.
“Structural Gain”

- $M$ user Gaussian multiple-access channel.

- Can use computation codes to efficiently decode the sum of codewords.

- Diminishing rates for standard strategies (for decoding the sum).

$$R_{\text{COMP}} = MR_{\text{LEGACY}}$$
Beyond Distributed Computation

- Now we know that computation codes are useful for reliable distributed computation.
- What does this have to do with communicating bits?
- Computation codes provide a coding strategy for exploiting interference.
A (Simple) Distributed MIMO Example

The “Sum-Difference” Relay Channel:

- Two senders, two relays, and one receiver.
- One relay sees the sum and the other sees the difference.
Strategy 1: Standard Routing

- Each relay **decodes** one of the transmitted codewords and forwards it towards the destination.
  
  Relay 1 decodes $X_1$
  Relay 2 decodes $X_2$

- This strategy is fundamentally **interference-limited** as we must treat other overheard codewords as **noise**.

- Gives up on the MIMO gain.
Strategy 2: Quantize-and-Forward

- If the destination had both the sum and the difference it could easily recover the original codewords.

- Have each relay quantize its signal and send it to the destination for processing and decoding.

  Relay 1 quantizes $X_1 + X_2 + Z_1$
  Relay 2 quantizes $X_1 - X_2 + Z_2$

- This strategy is fundamentally noise-limited since we send both signal and noise.

- Benefits from MIMO gain.
Strategy 3: Compute-and-Forward

- Have one relay \textit{decode} the sum and the other \textit{decode} the difference with a \textit{computation code}.

  \begin{align*}
  \text{Relay 1 decodes } X_1 + X_2 \\
  \text{Relay 2 decodes } X_1 - X_2
  \end{align*}

- Not limited by \textit{noise} or \textit{interference}.

- Benefits from \textit{MIMO gain}. 
“Structural Gain”

- Compute-and-forward is the dominant strategy starting from intermediate SNR.

- Quantize-and-forward does well here but performance degrades with retransmission.

- What about beyond this special channel matrix?
Oblivious Computation with Fading

- In general, wireless transmissions will be seen as a noisy linear mixture not sums and differences.

- Computation codes can be generalized to this setting if the receivers know the channel coefficients (transmitters can be oblivious).

- Relays just decode a linear function of codewords based on rounded versions of the channel coefficients.

\[
Y_1 = 1.1X_1 - 2.3X_2 + Z_1 \quad \text{decodes} \quad X_1 - 2X_2
\]
\[
Y_2 = 5.4X_1 + 3.7X_2 + Z_2 \quad \text{decodes} \quad 5X_1 + 4X_2
\]
New Physical Layer Abstractions

- With these new tools in hand, we can revise the standard abstraction of the physical layer.

- **Standard**: Transform physical layer into a set of reliable bit pipes between users.

- **Compute-and-Forward**: Transform physical layer into a set of reliable linear equations according to the interference. Receivers try to get a full rank set of equations about their desired codewords. Enables cooperation while preserving modularity.

- This resembles network coding but here the equations comes from the wireless medium (instead of combining packets at the transmitter).
Example Abstraction

AWGN Network

\( w \rightarrow \hat{w} \)
Example Abstraction
Example Abstraction

\[ w \rightarrow \hat{w} \]
Example Abstraction
Example Abstraction
Implementation Issues

- Clearly, synchronization is an issue. Can either maintain good synchronization or search for computation codes with appropriate robustness.

- Good (but not perfect) channel knowledge is needed at the receivers.

- Code must have an appropriate linear structure. In practice, one could achieve this by combining linear constellations (i.e. rectangular QAM) with good linear codes (i.e. LDPC codes).

- Many more issues...
Concluding Remarks

- Computation codes provide a nice framework for \textit{distributed linear processing} with built-in error-correction.

- We have shown in previous work that (in theory) these codes can provide boosts for many cooperative strategies including distributed MIMO and wireless network coding, even at moderate SNR.

- Hopefully, these gains are possible in practice too...