Trapping Sets and LDPC Decoder Performance

October 15, 2008

Shiva Planjery, Dariush Divsalar, Kenneth Andrews, Sam Dolinar, and Jon Hamkins
LDPC Codes/Decoders have Trapping Sets

Definition:
• When an LDPC decoder fails, the error set $T$ is the collection of code symbols (variable nodes) that do not converge correctly.
• $T$ is an $(a,b)$ trapping set, where there are $a$ variable nodes in $T$, and $b$ check nodes have an odd number of connections to $T$.

Informally, a trapping set is a set of variable nodes that is not well connected to the rest of the graph, in a way that causes the decoder trouble.

Properties:
• Trapping sets depend on the decoder’s parity check matrix, and on the decoding algorithm.
• A good LDPC code has large $d_{\text{min}}$, so its error floor is determined by trapping sets.
• How to reduce the effects of trapping sets is open research. Some strategies:
  • Modify the parity check matrix (without changing the code)
  • Modify the decoding algorithm

Example:
If these three variable nodes are wrong, only one check node is unsatisfied. This is a $(3,1)$ trapping set.
Trapping Sets and LDPC Decoder Performance

Trapping Sets and the AR4JA (1280,1024) LDPC Code

• LDPC codes with short blocklength, high code rate, and low decoding threshold are particularly susceptible to trapping sets.
• The AR4JA (1280,1024) LDPC code is no exception.

Example:
• “Standard” belief propagation decoder, min* algorithm
• Its most common failure is caused by this (4,2) trapping set:
• Additional trapping sets observed:
Decoder Modifications to Eliminate 4-cycles

Hypothesis: Particularly problematic trapping sets involve 4-cycles:

Solution 1: Modify decoder to decode 4-cycle structure optimally

\[
\begin{align*}
\lambda_1^o &= \lambda_2^i + \ln \frac{1 + e^{(\lambda_3^i + \lambda_4^i)}}{e^{\lambda_3^i} + e^{\lambda_4^i}} \\
\lambda_2^o &= \lambda_1^i + \ln \frac{1 + e^{(\lambda_1^i + \lambda_4^i)}}{e^{\lambda_1^i} + e^{\lambda_4^i}} \\
\lambda_3^o &= \ln \frac{1 + e^{(\lambda_1^i + \lambda_2^i + \lambda_4^i)}}{e^{\lambda_1^i} + e^{\lambda_2^i + \lambda_4^i}} \\
\lambda_4^o &= \ln \frac{1 + e^{(\lambda_1^i + \lambda_2^i + \lambda_3^i)}}{e^{\lambda_2^i + \lambda_3^i}}
\end{align*}
\]

Nodes are split so that optimal MAP decoding can be done on the edges of the 4-cycle.
Decoder Modifications to Reduce Trapping Set Errors

Solution 2: Modify the parity check graph to eliminate 4-cycles

Interestingly, these two solutions are equivalent, except for the message-passing schedule.
Trapping Sets and LDPC Decoder Performance

Decoder Modifications to Reduce Convergence Rate

Hypothesis: By analogy to simulated annealing, perhaps the decoder should be forced to converge more slowly, to reduce the probability of finding false minima.

Solution 3: Modify the variable node update rule to include a damping factor:

\[ \beta^* = \sum_{i=1}^{d_v} \lambda_i^k + \lambda_c \]  
Traditional equation

\[ \beta^k = \gamma \cdot \beta^{k-1} + (1 - \gamma) \cdot \beta^* \]  
Modified equation

Solution 4: Modify the check node update rule to use \( \min \) instead of \( \min^* \), after some number of iterations have elapsed.

Solution 5: Combine solutions 3 and 4 to control the “cooling rate” of the decoder as it converges to a steady state.

Simulation results: These techniques do lower the error floor caused by trapping sets, but at a cost in threshold. They also increase the number of iterations required.
Trapping Sets and LDPC Decoder Performance

Decoder Modifications to Modify Stability

Hypothesis: If decoding stops when a codeword is found, then it doesn’t matter if the decoding algorithm is stable afterwards. By making the algorithm unstable, perhaps it will depart from trapping sets and keep searching for the correct solution.

Solution 6: Modify the variable node equation so message magnitudes do not grow exponentially as decoder converges.

Simulation results: This technique lowers the error floor nearly two decades with no penalty in threshold.