Analyzing the Peeling Decoder

Supplemental Material for Advanced Channel Coding

Henry D. Pfister

January 25th, 2012

1 Introduction

The simplest example of iterative decoding is the peeling decoder introduced for the binary erasure channel (BEC) by Luby et al. [1] (see also [2, pp. 117–121, 134–136]) . This decoder is based on the parity-check matrix of the code and can be applied to an arbitrary linear code[1]. Let H be the parity-check matrix of an (n, k) linear code.

Definition 1.1. The *Tanner graph* G of an $m \times n$ parity-check matrix is a bipartite graph with one vertex for each code symbol and one vertex for each parity check. For each non-zero entry in H, the graph contains an edge that connects the variable node associated with the column to the check node associated with the row. Mathematically, we have $G = (V \cup C, E)$ with

$$V = \{1, 2, \dots n\} \qquad C = \{1, 2, \dots, m\} \qquad E = \{(i, j) \in C \times V \,|\, \text{if } H_{i,j} \neq 0\}\,.$$

Algorithm 1.2 (Peeling Decoder)**.** *Iteratively remove known bits from the graph as follows:*

1. *Initialize the variables $x_1, \dots, x_n$ to ? and the variables $y_1, \dots, y_m$ to zero.*
2. *For each non-erased code symbol, let $j \in V$ be its index and set variable x_j to the known value.*
3. *If there is a degree-1 check node, let $j \in C$ be its index, $i \in V$ be the index of the adjacent variable node, and set $x_j = H_{ij}^{-1} y_i$.*
4. *If the graph contains a variable node whose value is known (i.e., $x_j \neq ?$), let $j \in V$ be its index and*
   (a) *for all i such that $(i, j) \in E$, update $y_i = y_i - H_{ij} x_j$*
   (b) *remove bit j and all adjacent edges from the graph (i.e., $V \leftarrow V \backslash j$, $E \leftarrow E \backslash \{(i, j') \in E \,|\, j = j'\}$)*
   (c) *Goto step 3*
5. *When the algorithm reaches this point, either decoding is successful and $x_j \neq ?$ for all $j \in V$ or the decoder is stuck in a configuration where there are no degree-1 check nodes and the graph contains only variable nodes whose values are unknown.*

Exercise 1.3. Implement the peeling decoder in some programming language (e.g., Matlab) and test it on the $(7, 4)$ Hamming code with parity-check matrix

$$H = \begin{bmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}. \tag{1}$$

[1] While our primary interest is binary codes, this description is valid for codes defined over any field.

2 Static Analysis of the Peeling Decoder

For any parity-check matrix, the performance of the peeling decoder is completely determined by the stopping sets of its Tanner graph.

Definition 2.1 (Stopping Set)**.** A *stopping set* (ss) S is a subset of variable nodes that, when initially erased, prevents the peeling decoder from recovering the value of any variable node in S. Mathematically, S is a subset of variable nodes whose induced subgraph has no check nodes with degree 1 (i.e., all neighbors of S must be connected to S at least twice). For example, to satisfy the parity-check equations, the subgraph induced by the support set of any codeword cannot have any degree-1 check nodes and therefore must be a stopping set. By convention, the empty set is considered a stopping set.

Lemma 2.2 (Lemma 3.140 in [2])**.** *Stopping sets have the following properties:*

1. *If S_1 and S_2 are ss, then $S_1 \cup S_2$ is a ss.*
2. *Each subset W of V contains a unique maximum ss.*
3. *If $W \subseteq V$ is the set of initially erased variable nodes, then the peeling decoder will recover all variables except those in the unique maximum ss contained in W.*

Proof. For the first claim, we observe that, if a check node c is a neighbor of $S_1 \cup S_2$, then it must be a neighbor of either S_1 or S_2. Since S_1 and S_2 are both stopping sets, either S_1 or S_2 must be connected to c at least twice. Therefore, $S_1 \cup S_2$ is connected to c at least twice.

For the second claim, we define U to be the union of all stopping sets contained in W. The first claim implies that U is indeed a ss. It is maximal because any ss S contained in W (i.e., $S \subseteq W$) is also contained in U (i.e., $S \subseteq U$).

For the third claim, we first observe that, if W contains a stopping set S (i.e., $S \subseteq W$), then the peeling decoder will not recover any variable in S. This follows from the definition of a ss and is based on the fact that even revealing all variables $V \backslash S$ does not allow the peeling decoder to recover any variable in S. This also implies that the peeling will not recover any variables in U, the unique maximum ss contained in W. Since U is the maximum ss in W, for all $T \subseteq A$, the set $T \cup U$ is not a stopping set and therefore will be reduced by one step of the peeling decoder. This implies that all variables in $W \backslash U$ will be recovered by the peeling decoder. □

Definition 2.3. A ss is *minimal* if the only stopping set it contains is the empty set.

For a particular parity-check matrix, let $A_{ss}(s)$ be the number of stopping sets of weight h and $\hat{A}_{ss}(h)$ be the number of minimal stopping sets of weight h. For the BEC, one can use $\hat{A}_{ss}(h)$ to bound the probability that the peeling decoder does not successfully recover all symbols. Since a decoding failure occurs only if the set of channel erasures contain some minimal ss, one has

$$P_B(\epsilon) \leq \sum_{h=1}^{n} \hat{A}_{ss}(h)\epsilon^h. \tag{2}$$

Problem 2.4. Can the exact decoding failure probability be computed in solely terms of the minimal stopping weight enumerator $\hat{A}_{ss}(h)$? [Hint: Try finding a code with the same $\hat{A}_{ss}(h)$ and different $\hat{A}_{ss}(h)$].

Exercise 2.5. Use your program from Exercise 1.3 to find all stopping sets for the peeling decoder with (1) and list the minimal stopping sets. Simulate this code/decoder on the BEC and compare the performance with 2. Compare the performance on One can also apply the peeling decoder to an overcomplete parity-check matrix whose rows have linear dependencies. Repeat this exercise for the

same code with the overcomplete parity-check matrix

$$H = \begin{bmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}. \tag{3}$$

3 Dynamic Analysis of the Peeling Decoder

While one can always use simulations to observe the dynamics of complex systems (e.g., the peeling decoder), this approach does not typically lead to simple design principles. If one analyzes the peeling decoder for a randomly-chosen code instead, then the analysis is greatly simplified. The overall result is a set of design rules that provide considerable insight.

3.1 Code Ensembles

An *ensemble* of codes is a probability distribution over a set of codes. In most cases, an ensemble is defined by a probability distribution over either the set of generator matrices or the set of parity-check matrices. For iterative decoding, this subtle difference is important because the decoding performance depends not only on the code but also on its representation.

Example 3.1. Let $\mathcal{P}(n, k, \mathtt{l})$ be the Poisson code ensemble defined by choosing uniformly from the set of $(n-k) \times n$ parity-check matrices with exactly $\mathtt{l}$ ones in each column. One can draw a random parity-check matrix from this ensemble simply by choosing each column of the parity-check matrix uniformly from the set of binary vectors with exactly $\mathtt{l}$ ones.

As we saw in Definition 1.1, every parity-check matrix can be mapped to a Tanner graph. Likewise, every Tanner graph can be mapped back into a parity-check matrix. Therefore, an ensemble of codes defined by parity-check matrices is naturally interchangeable with an ensemble of codes defined by Tanner graphs. This allows one to define ensembles of codes using ensembles of Tanner graphs.

Example 3.2. Let $\mathcal{R}(n, \mathtt{l}, \mathtt{r})$ be the regular code ensemble, for $n \in \mathtt{r}\mathbb{N}$, defined by an ensemble of Tanner graphs chosen as follows. First, define n variable nodes, each with $\mathtt{l}$ edge *sockets,* and label the sockets from 1 to $n\mathtt{l}$. Then, define $n\mathtt{l}/\mathtt{r}$ check nodes, each with $\mathtt{r}$ edge sockets and label the edge sockets from 1 to $n\mathtt{l}$. Next, pick a uniform random permutation σ on $n\mathtt{l}$ elements. The construction is completed by attaching, for $i = 1, \ldots, n\mathtt{l}$, the i-th variable node socket to the $\sigma(i)$-th check node socket. Since the resulting graph can have multiple edges connecting the same vertices, this defines a random bipartite multigraph. Mapping the Tanner graph back to a parity-check matrix also collapses multiple edges: to a single edge if the multiplicity is odd and to nothing if the multiplicity is even.

A similar approach can be used to define the *standard ensemble* LDPC(Λ, P) of irregular low-density parity-check (LDPC) codes using an ensemble of Tanner graphs. Let Λ_i be the number of variable nodes with degree i and define $\Lambda(x) = \sum_{i=1}^{\mathtt{l}_{\max}} \Lambda_i x^i$. Likewise, let P_i be the number of check nodes with degree i and define $P(x) = \sum_{i=1}^{\mathtt{r}_{\max}} P_i x^i$. The polynomial $\Lambda(x)$ (resp. $P(x)$) is called the variable (resp. check) *degree distribution from the node perspective.* In terms of these, one can compute the block length $n = \Lambda(1)$, the number of checks $m = P(1)$, and the number of edges in the graph $e = \Lambda'(1) = P'(1)$.

Definition 3.3 (The Standard Ensemble LDPC(Λ,P))**.** Like the regular ensemble $\mathcal{R}(n, \mathtt{l}, \mathtt{r})$ above, the standard irregular ensemble is constructed using a random permutation to connect variable node sockets to check node sockets. For $i = 1, \ldots, \mathtt{l}_{\max}$, we define Λ_i variable nodes, each with i edge *sockets,* and then label all sockets from 1 to e. For $i = 1, \ldots, \mathtt{r}_{\max}$, define P_i check nodes, each with i edge sockets, and then label all the edge sockets from 1 to e. Next, pick a uniform random permutation σ on e elements. The construction is completed by attaching, for $i = 1, \ldots, e$, the i-th variable node socket to the $\sigma(i)$-th check node socket.

3.2 Markov Chain Analysis

Suppose a random code is drawn from the LDPC(Λ, P) ensemble and the all-zero codeword is transmitted over a BEC. Decoding starts by first peeling off known bits that were not erased by the channel. Then, decoding continues by using degree-1 check nodes to reveal bits one at a time. In this case, the state of peeling decoder is captured entirely by the Tanner graph. In fact, the sequence of Tanner graphs forms a Markov process. Unfortunately, the number of Tanner graphs is extremely large and analyzing this Markov chain directly is infeasible.

It turns out the that state space of the Markov chain can be reduced significantly. The key observation is that the edges of the Tanner graph can revealed only as they are needed. In this case, the connections between edges that have not been revealed are still governed by a uniform random permutation. Therefore, the state of the Markov chain is captured entirely by the graph's degree distribution.

Lemma 3.4. *Let G be a random graph drawn from the LDPC(Λ, P) ensemble. Suppose a node is chosen in a way that does not depend on the edge connections and removed along with all of its edges. Then, the conditional distribution of the resulting resulting graph G', given the degree and type of the removed node, is given by the LDPC(Λ', P') ensemble, where (Λ', P') is degree distribution of G'.*

Sketch of Proof. The idea is to focus on the permutation that connects the bit and check nodes. If one deletes a degree-i node and removes its edges, then i values are removed from the permutation. If the original permutation was uniform on e elements, then the resulting permutation is uniform on $e - i$ elements. □

The Markov chain on the reduced state space is now described. Let the r.v. $X_{k,i}$ (resp. $Y_{k,i}$) be the number of variable (resp. check) nodes with degree i after k edges have been removed. The implied sequence of degree distribution vectors are denoted $X_k = (X_{k,1}\ X_{k,2}\ \dots\ X_{k,\mathtt{l}_{\max}})$ and $Y_k = (Y_{k,1}\ Y_{k,2}\ \dots\ Y_{k,\mathtt{r}_{\max}})$. Suppose one wants to remove a variable node and all its attached edges. One can do this by immediately removing the node and then removing the remaining half-edges one at a time. Consider this process for a variable node of degree i starting at the k-th step (i.e., after k other edges have been removed), then the deterministic effect for the node removal is $X_{k+l,i} = X_{k,i} - 1$ for $l = 1, \dots, i$. But, there is also a random effect on Y_k because the half-edges are mapped to the check nodes through a random permutation. The effect is equivalent to to sequentially deleting i randomly chosen check edges. For the l-th edge, the degree of the attached check node is chosen randomly according to

$$p(j) = \frac{Y_{k+l,j} j}{\sum_{j'} Y_{k+l,j'} j'},$$

for $l = 1, \dots, i$. The deterministic effect of deleting the l-th check edge (from a check node of degree j) is $Y_{k+l,j} = Y_{k+l-1,j} - 1$ and $Y_{k+l,j-1} = Y_{k+l-1,j-1} + 1$ (i.e., a degree j check node is replaced by a degree $j - 1$ check node).

Using these equations, one can simulate the decoder for a randomly chosen code and erasure pattern without ever generating the code or graph. The idea is to initialize the X_0, Y_0 vectors and then randomly update these vectors according the stochastic peeling process. In the first stage, the node removal process above is repeated for each bit that is not erased by the channel. In the second stage, degree-1 checks are used to recover unknown bits which are then removed. The degree of the recovered bit is chosen randomly according to

$$q(j) = \frac{X_{k,j} j}{\sum_{j'} X_{k,j'} j'}$$

and its removal causes the update $X_{k+l,i} = X_{k,i} - 1$ for $l = 1, \dots, i$. On the check node side, one degree-1 node is removed deterministically (i.e., $Y_{k+1,1} = Y_{k,1} - 1$) and $j - 1$ edges are removed randomly as described above. The process either terminates with all nodes are removed or in a stopping set with no degree-1 check nodes.

Remark 3.5. If the channel introduces e erasures, then applying the first stage of known-bit removal to a randomly chosen code from $\mathcal{P}(n, k, \mathtt{1})$ results in a random code from $\mathcal{P}(e, e - n + k, \mathtt{1})$. Caveat: even though $e - n + k$ is generally negative, the ensemble is well-defined because each matrix has $e - (e - n + k) = n - k$ rows. Therefore, analyzing the peeling decoder for this ensemble can be reduced to the case where all bits are erased.

References

[1] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. A. Spielman, "Efficient erasure correcting codes," *IEEE Trans. Inform. Theory*, vol. 47, pp. 569–584, Feb. 2001.

[2] T. J. Richardson and R. L. Urbanke, *Modern Coding Theory*. New York, NY: Cambridge University Press, 2008.