The Analysis of Graph Peeling Processes

Supplemental Material for Graphical Models and Inference
Henry D. Pfister

October 27th, 2014

1 Introduction

Random graphs are now a standard tool in computer science, engineering, and mathematics. A variety of questions about random graphs can be answered by analyzing the average peformance of simple graph peeling algorithms [1]. In general, the intial graph is random and simple rules are used to sequentially remove edges and vertices until none of the rules can be applied. Often one is interested in the size of the residual graph as a function of some parameter.

2 The Peeling Decoder for LDPC Codes

The simplest example of iterative decoding is the peeling decoder introduced for the binary erasure channel (BEC) by Luby et al. [2] (see also [3, 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 2.1. 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 2.2 (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.
```

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

Exercise 2.3. Implement the peeling decoder in some programming language (e.g., Matlab) and test it on the $(7,4)$ Hamming code with parity-check matrix

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

3 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 3.1 (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 3.2 (Lemma 3.140 in [3])**.** *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 3.3. 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

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

Problem 3.4. Can the exact decoding failure probability be computed in solely terms of the minimal stopping weight enumerator $\hat{A}_{ss}(h)$? [Hint: Try finding two codes with the same $\hat{A}_{ss}(h)$ but different $P_B(\epsilon)$].

Exercise 3.5. Use your program from Exercise 2.3 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

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

4 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.

4.1 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 4.1. 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 2.1, 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 4.2. 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 4.3 (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.

4.2 Markov Graph Process

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 4.4. *Let G be a random graph drawn from the LDPC(Λ, P) ensemble. Suppose a node is chosen randomly in a way that depends only on its degree. Then, its edges and the degrees of its adjacent nodes are revealed. Let G' be the graph resulting from removing the chosen node along with all of its edges and let (Λ', P') be its degree distribution. Then, conditional on the revealed information, G' is a uniform random graph from the LDPC(Λ', P').*

Sketch of Proof. The idea is to focus on the degree distributions and the permutation that connects the bit and check nodes. If one deletes a degree-i node and reveals both its edges and the degrees of its adjacent nodes, then (Λ', P') is a deterministic function of the revealed information and (Λ, P). Also, deleting the node's edges eliminates i edge sockets and i values of the permutation. Since the surviving values of the permutation are independent of the revealed information, they remain uniform on the surviving $e - i$ elements. $\square$

A simulation strategy is now described for the Markov chain on the reduced state space. 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 $X_k = (X_{k,1}\ X_{k,2}\ \ldots\ X_{k,\mathtt{l}_{\max}})$ and $Y_k = (Y_{k,1}\ Y_{k,2}\ \ldots\ Y_{k,\mathtt{r}_{\max}})$. 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 $l = 1, \ldots, i$. 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 $l = 1, \ldots, i$. 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 $l = 1, \ldots, i$. 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 4.5. 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.

5 Analysis Based on an Urn Model

Consider a urn initially filled with n colored balls where each color is denoted by an integer $1, \ldots, C$. The composition of the urn is modified sequentially based on the follow rule:

> A ball is drawn uniformly from the urn. Based on the color of the randomly chosen ball, a number of other balls are deterministically added or removed from the urn.

Let the r.v. X_i^j be the number of balls with color j after i steps. The state of the urn after i steps is therefore given by the vector $X_i = \left(X_i^1, X_i^2, \ldots, X_i^C\right)$. If a ball of color of k is drawn, then $[A]_{jk} = a_{jk}$ balls of color j are added to the urn. The probability of picking a ball with color j during the ith step is given by $X_i^j / \|X_i\|$, where $\|X_i\| \triangleq \sum_{j=1}^{C} \left|X_i^j\right|$ is the total number of balls in the urn after i steps. Therefore, the number of balls in the urn satisfies the stochastic recursion

$$X_{i+1} = X_i + A\, Z_i, \tag{4}$$

where $Z_i = e_j$ (i.e., all zero column vector except for a one in row j) with probability $X_i^j / \|X_i\|$. This process is well-defined as long as $X_i^j \geq 0$ for $j = 1, \ldots, C$.

Much of the modern interest in urn models dates back to 1923 paper by Eggenberger and Polya which introduced the simple model now known as *Polya's Urn*. The generalization described above has been analyzed thoroughly (e.g., strong law, central-limit, large deviations, exact m.g.f.) in the case where the initial condition is fixed and the number of balls is non-decreasing (i.e., $A\mathbf{1} \succeq \mathbf{0}$) [4]. In this work, we focus on an example where the number of balls is strictly decreasing but the process starts with a large number of balls.

5.1 Mean Behavior

As the number of balls in the urn becomes large, a scaled version of this process tends a deterministic limit. Consider the scaled process $X(t) = \lim_{n\to\infty} \frac{1}{n} X_{\lfloor nt \rfloor}$. While it is not immediately clear that this limit exists, one can show that the derivative

$$X'(t) \triangleq \lim_{\epsilon\to 0} \lim_{n\to\infty} \frac{\frac{1}{n} X_{\lfloor n(t+\epsilon)\rfloor} - \frac{1}{n} X_{\lfloor nt \rfloor}}{\epsilon}$$

approaches a deterministic limit with probability 1. The reason is that $X_{\lfloor n(t+\epsilon)\rfloor} - X_{\lfloor nt \rfloor}$ equals the sum of $n\epsilon$ almost i.i.d. random vectors whose common distribution defines the drift. As $\epsilon \to 0$, these vectors become i.i.d. and their average converges to the expected drift $A\, X(t) / \|X(t)\|$. The function $X(t)$ is called the fluid limit of the process and theorems establishing this convergence were developed by Kurtz in the 1970's [5]. They were also used in the 1990's to solve problems in computer science and coding by Wormald and Mitzenmacher [1, 6].

For a process with a fluid limit, the behavior is essentialy determined by the scaled mean, $\overline{X}(t) = \lim_{n\to\infty} \mathbb{E}\left[\frac{1}{n} X_{\lfloor nt \rfloor}\right]$. For the general urn model, $\overline{X}(t)$ satisfies the differential equation

$$\overline{X}'(t) = \frac{A\, \overline{X}(t)}{\|\overline{X}(t)\|}, \tag{5}$$

which has a well-defined solution for $0 \le t < T$, where T is largest real number such that all elements of $\overline{X}(t)$ are non-negative. The mean process simplifies when the columns sums of A are constant (i.e., $\sum_j a_{jk} = q$ for all $1 \le k \le C$) because this implies that $\|X_i\| = n + qi$ is deterministic. Under this assumption, one the mean satisfies[2]

$$\begin{aligned}
\mathbb{E}\left[X_{i+1}\right] &= \mathbb{E}\left[X_i\right] + \mathbb{E}\left[A\, Z_i\right] \\
&= \mathbb{E}\left[X_i\right] + \frac{A}{n+qi}\mathbb{E}\left[X_i\right] \\
&= \left(I + \frac{A}{n+qi}\right)\mathbb{E}\left[X_i\right] \\
&= \left[\prod_{j=0}^{i}\left(I + \frac{A}{n+q(i-j)}\right)\right]\mathbb{E}\left[X_0\right].
\end{aligned}$$

For this reason, we define the matrix $L_i = \prod_{j=0}^{i}\left(I + \frac{A}{n+q(i-j)}\right)$ and observe that

$$\begin{aligned}
L_{\lfloor nt \rfloor} &= \prod_{j=0}^{\lfloor nt \rfloor}\left(I + \frac{A}{n+q(\lfloor nt \rfloor - j)}\right) \\
&= \prod_{j=0}^{\lfloor nt \rfloor}\left[\left(I + O\left(\frac{1}{n^2}\right)\right)\exp\left(\frac{1}{n(1+qt)-qj}A\right)\right] \\
&= O\left(\frac{1}{n}\right) + \prod_{j=0}^{\lfloor nt \rfloor}\exp\left(\frac{1}{n(1+qt)-qj}A\right) \\
&= O\left(\frac{1}{n}\right) + \exp\left(A\sum_{j=0}^{\lfloor nt \rfloor}\frac{1}{n(1+qt)-qj}\right) \\
&= O\left(\frac{1}{n}\right) + \exp\left(A\int_0^t \frac{1}{1+qx}dx + O\left(\frac{1}{n}\right)\right),
\end{aligned}$$

where a matrix function $f : \mathbb{N} \to \mathbb{R}^{C\times C}$ satisfies $f(n) = O(g(n))$ for $g : \mathbb{N} \to \mathbb{R}$ if the induced matrix norm satisfies $\|f(n)\| = O(g(n))$. For example, the third equality follows from

$$\begin{aligned}
Q &= \left\|\prod_{j=0}^{\lfloor nt \rfloor}\left[\left(I + O\left(\frac{1}{n^2}\right)\right)\exp\left(\frac{1}{n(1+qt)-qj}A\right)\right] - \prod_{j=0}^{\lfloor nt \rfloor}\exp\left(\frac{1}{n(1+qt)-qj}A\right)\right\| \\
&\le \left[\left(1 + O\left(\frac{1}{n^2}\right)\right)^{\lfloor nt \rfloor} - 1\right]\prod_{j=0}^{\lfloor nt \rfloor}\left\|\exp\left(\frac{1}{n(1+qt)-qj}A\right)\right\| \\
&\le \left(1 + O\left(\frac{1}{n}\right)\right)\prod_{j=0}^{\lfloor nt \rfloor}\exp\left(\frac{1}{n(1+qt)-qj}\,\|A\|\right) \\
&= \left(1 + O\left(\frac{1}{n}\right)\right)\exp\left(\|A\|\sum_{j=0}^{\lfloor nt \rfloor}\frac{1}{n(1+qt)-qj}\right) \\
&\le O\left(\frac{1}{n}\right)\left(\exp\left(\|A\|\int_0^t \frac{1}{1+qx}dx\right)dt + O\left(\frac{1}{n}\right)\right) \\
&= O\left(\frac{1}{n}\right).
\end{aligned}$$

[2]Since the product of matrices is non-cummutative, we must adopt some convention for $\prod_{i=a}^{b} A_i$. Thus, we assume that products are composed from left to right and $\prod_{i=a}^{b} A_i = A_a \ldots A_b$ for $a \le b$.

This implies that, for $t \in (0,T)$,

$$\lim_{n\to\infty} L_{\lfloor nt \rfloor} = e^{A\int_0^t \frac{1}{1+qx}dx} = e^{A\frac{1}{q}\ln(1+qt)}$$

and

$$\overline{X}(t) = \lim_{n\to\infty} \frac{1}{n}\mathbb{E}\left[X_{\lfloor nt \rfloor}\right] = \lim_{n\to\infty} L_{\lfloor nt \rfloor}\frac{1}{n}\mathbb{E}\left[X_0\right] = e^{A\frac{1}{q}\ln(1+qt)}\overline{X}(0).$$

Exercise 5.1. Verify that the above equation for $\overline{X}(t)$ satisfies the differential equation (5).

5.2 Connection to LDPC Codes

Consider a bit-regular LDPC code transmitted over a binary erasure channel (BEC) and decoded with a peeling style decoder. Assume that all bit nodes have degree d and that the code initially has with n edges (i.e., it has n/d bit nodes). The balls in the urn model will represent edges in the LDPC decoding graph and the color of a ball will denote the degree of the check node adjacent to the edge. Therefore, the check degree distribution is defined by the number of edges X_0^j initially attached to degree j check nodes for $j = 1, 2, \ldots, C$.

Decoding proceeds in two stages that can both be represented by the above urn model.

In the first stage, correct (i.e., non-erased) observations from the channel are handled. Assume that the channel erases a uniform random pattern of exactly pn/d bits. This means that $(1-p)n/d$ bits are observed correctly from the channel. The peeling decoder uses each observation to remove all edges attached to that bit and merge the observed value into the attached check nodes. On the check node side, this is equivalent to randomly removing exactly $(1-p)n$ edges. If an edge attached to a degree j check node is removed, then this reduces the degree of the check node by 1 and results in j edges of degree j being transformed into $j-1$ edges of degree $j-1$. In the urn model, j balls of color j are replaced by $j-1$ balls of color $j-1$ and the resulting random process can be constructed by applying (4) exactly $(1-p)n$ times using the matrix

$$A = \begin{bmatrix} -1 & 1 & & & & \\ & -2 & 2 & & & \\ & & -3 & \ddots & & \\ & & & \ddots & C-2 & \\ & & & & -(C-1) & C-1 \\ & & & & & -C \end{bmatrix}.$$

After the first stage, the number of edges that are adjacent to check nodes of degree j is given by $X_{(1-p)n}^j$. In the second stage, decoding proceeds by using degree-1 check nodes to determine the values of erased bit nodes. Each recovered bit node causes the removal of a single degree-1 check edge and $d-1$ randomly chosen check edges. This leads to one deterministic step in the urn model and $d-1$ random steps using the matrix A. Therefore, for $i \geq (1-p)n$, the random process is defined by

$$X_{i+1} = \begin{cases} X_i - e_1 & \text{if } i \bmod d = 0 \\ X_i + A\,Z_i & \text{otherwise,} \end{cases}$$

where e_j is the jth standard unit vector. Since this process is well-defined only as long as $X_i^1 \geq 0$, we let $N = \min\left\{i \in \mathbb{N} \,|\, X_i^1 = 0,\, i \geq (1-p)n\right\}$ the stopping time of the process. This process runs until decoding is finished at time $N = n$ or stops early due to a lack of degree-1 check nodes at time $N < n$.

For the LDPC case, $q = -1$ and the first phase of decoding can be characterized precisely by

$$\overline{X}(t) = e^{-A\ln(1-t)}\overline{X}(0), \quad 0 \leq t \leq 1-p.$$

Exercise 5.2. This differential equation actually has the closed-form solution

$$\overline{X}^j(t) = \sum_{k=0}^{C} \overline{X}^k(0)\frac{j}{k}\binom{k}{j}t^{k-j}(1-t)^j.$$

Verify that this solution satisfies the differential equation (5). Can you give a short probabilistic proof for this expression?

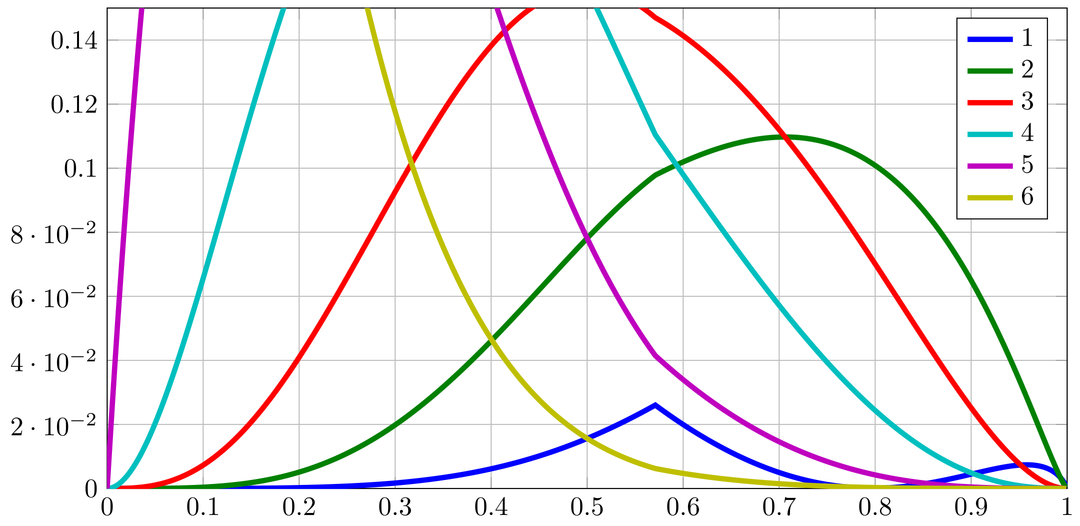

Figure 1: For the regular LDPC ensemble $\mathcal{R}(n, \mathtt{l}, \mathtt{r})$ with $\mathtt{l} = 3$ and $\mathtt{r} = 6$, the solution to the above differential equation is shown for $p = 0.429$. This p is just below the maximum erasure threshold $p^* = 0.4294$ that allows successful decoding. Thus, the fraction of degree-1 edges is strictly positive for $t \in (0, 1)$. The change from the first phase to the second phase at $t = 0.571$ can be seen clearly from the slope discontinuity in the fraction of degree-1 edges.

For the second phase of decoding (i.e., $t > 1 - p$), the update alternates between using A and removing one degree-1 edge. Let the matrix B be zero except for the first row, where each entry equals -1. Then, $BZ_i = -e_1$. In the above calculation, the effect of using B instead of A for every d-th step is asymptotically equivalent to using the average

$$\tilde{A} \triangleq \frac{1}{d} B + \frac{d-1}{d} A.$$

Thus, we can use the above approach to integrate starting from $1 - p$ and the second phase of decoding is characterized precisely by

$$\overline{X}(t) = e^{-\tilde{A} \ln\left(\frac{1-t}{p}\right)} \overline{X}(1-p), \quad 1 - p \le t \le T.$$

These equations are plotted in Figure 1 for the regular LDPC ensemble $\mathcal{R}(n, \mathtt{l}, \mathtt{r})$ with $\mathtt{l} = 3$ and $\mathtt{r} = 6$.

6 General Approach

The urn model described in the last section is just one example of a system whose normalized large-n behavior beomes deterministic and satisfies a differential equation. In the general case, for each $n \in \mathbb{N}$, consider a sequence of random vectors $X_i \in \mathbb{R}^C$ for $i = 0, 1, \ldots, M(n)$ where:

1. For a bounded set $D \subseteq \mathbb{R}^{C+1}$, there is a Lipschitz continuous $f = (f_1, \ldots, f_C) : D \to \mathbb{R}^C$ such that

$$\mathbb{E}\left[X_{i+1}^j - X_i^j \,|\, X_0, X_1, \ldots, X_i\right] = f_j\left(i/n, X_i^1/n, \ldots, X_i^C/n\right) + O\left(\frac{1}{n}\right)$$

2. $M(n) = \sup_{i \in \mathbb{N}} \left\{ i \in \mathbb{N} \,\middle|\, \left(i/n, X_i^1/n, \ldots, X_i^C\right) \in D \right\}$ is the stopping time for the process

3. For some $A < \infty$, $\left| X_{i+1}^j - X_i^j \right| \le A$ for all $j \in \{1, 2, \ldots, C\}$ and all $0 \le i \le M(n)$

4. $\lim_{n \to \infty} \frac{1}{n} X_0^j$ is almost surely constant for $j \in \{1, 2, \ldots, C\}$

Under these conditions, one finds that

1. There is a T such that the limit $X(t) \triangleq \lim_{n\to\infty} \frac{1}{n} X_{\lfloor nt \rfloor}$ exists and is deterministic for $0 \le t < T$
2. The trajectory $X(t)$ is the unique solution of the ordinary differential equation $X'(t) = f(t, X^1(t), \ldots, X^C(t))$ starting at $X(0)$, where $X^j(t) = [X(t)]_j$

For a concise proof of this, see [2, Appendix A].

This generalization allows one to analyze many greedy graph peeling algorithms. For example, consider the case of generalized LDPC (GLDPC) codes where the single-parity-check contraints are replaced by stronger code constraints that allow one to correct all patterns of $c > 1$ erasures. Consider a peeling decoder that proceeds by choosing uniformly a constraint node with degree $\le c$ and then correcting one of its adjacent erased bits. If the GLDPC code is bit-regular with bit-degree d, then the graph update consists of choosing a random constraint node of degree c or less and then random deleting one of its adjacent variable nodes and its d edges. Let X_i^j be the number of edges attached to degree-j constraints after i edges have been removed. For any $\alpha > 0$, the domain is $D = \left\{ (t, x_1, \ldots, x_C) \in \mathbb{R}_{\ge 0}^{C+1} \mid \sum_{j=1}^{c} x_j \ge \alpha \right\}$ and the j-th component of the drift is

$$f_j(t, x_1, \ldots, x_C) = \frac{1}{d} \sum_{k=1}^{c} a_{jk} \left(\frac{x_k/k}{\sum_{k'=1}^{c} x_{k'}/k'} \right) + \frac{d-1}{d} \sum_{k=1}^{C} a_{jk} \left(\frac{x_k}{\sum_{k'=1}^{C} x_{k'}} \right),$$

where a_{jk} denotes the entries in the A matrix for the LDPC urn model.

References

[1] N. C. Wormald, "Differential equations for random processes and random graphs," *Ann. Appl. Probab.*, pp. 1217–1235, 1995.

[2] 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.

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

[4] H. Mahmoud, *Pólya urn models.* CRC press, 2008.

[5] T. G. Kurtz, "Solutions of ordinary differential equations as limits of pure jump Markov processes," vol. 7, no. 1, pp. 49–58, 1970.

[6] M. Mitzenmacher, "Load balancing and density dependent jump Markov processes," in *Proc. IEEE Symp. on the Found. of Comp. Sci.*, pp. 213–222, 1996.