

Density Evolution for LDPC Codes on the BEC

Supplemental Material for Error Correcting Codes
Henry D. Pfister

January 23rd, 2022

1 Introduction

These notes are intended to provide a simple and intuitive introduction to density evolution for LDPC codes on the BEC. The presentation is based on [1] but adds a few examples and omits some details.

2 Preliminaries

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

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

Definition 2.2. 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.

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 2.3. Let $\mathcal{R}(n, 1, r)$ be the **regular code ensemble**, for $n \in \mathbb{rN}$, defined by an ensemble of Tanner graphs chosen as follows. First, define n variable nodes, each with 1 edge *sockets*, and label the sockets from 1 to $n1$. Then, define $n1/r$ check nodes, each with r edge sockets and label the edge sockets from 1 to $n1$. Next, pick a uniform random permutation σ on $n1$ elements. The construction is completed by attaching, for $i = 1, \dots, n1$, 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. To get a valid parity-check matrix, one must also collapse multiple edges. All multiple edges are replaced by a single edge if the edge multiplicity is odd. Otherwise, the edge is removed because all edges cancel if the edge multiplicity is even.

3 The Standard Irregular Ensemble

The standard ensemble LDPC(Λ, P) of irregular low-density parity-check (LDPC) codes is defined by an ensemble of Tanner graphs. Let Λ_i be the number of variable nodes with degree i and define $\Lambda(x) = \sum_{i=1}^{1\max} \Lambda_i x^i$. Likewise, let P_i be the number of check nodes with degree i and define $P(x) = \sum_{i=1}^{r\max} P_i x^i$. The polynomial $\Lambda(x)$ (resp. $P(x)$) is called the variable (resp. check) **unnormalized 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 3.1. A sample from the **standard irregular ensemble** LDPC(Λ, P) is constructed using a random permutation to connect variable node sockets to check node sockets. For $i = 1, \dots, \mathbf{l}_{\max}$, we define Λ_i variable nodes, each with i edge *sockets*, and then label all sockets from 1 to e . For $i = 1, \dots, \mathbf{r}_{\max}$, 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 $i = 1, \dots, e$, the i -th variable node socket to the $\sigma(i)$ -th check node socket. The symbol \mathbf{G} will be used to denote a random Tanner graph drawn from this ensemble.

Remark 3.2. The possibility of double edges does cause one subtle issue. For the code, the parity-check matrix is given by the biadjacency matrix of the graph formed by collapsing a multiple edge into a single edge if its multiplicity is odd and removing the multiple edge completely if its multiplicity is even. But, the iterative decoder does not automatically collapse these double edges and this can lead to suboptimal performance. In practice, this is not an issue because one typically modifies the construction to avoid all double edges and four cycles.

Exercise 3.3. Write a Python program that samples \mathbf{G} (e.g., by returning the sparse biadjacency matrix) for the **standard irregular ensemble** LDPC(Λ, P). Such a program could also compute the number of multiple edges and the number of edges involved in 4-cycles.

When the degree distributions are normalized to sum to one, they are denoted by $L(x) = \sum_{i=1}^{\mathbf{l}_{\max}} L_i x^i = \Lambda(x)/\Lambda(1)$ and $R(x) = \sum_{i=1}^{\mathbf{r}_{\max}} R_i x^i = P(x)/P(1)$, where L_i (resp. R_i) is the fraction of variable nodes (resp. check nodes) with degree i . These distributions are called the **normalized degree distribution from the node perspective** and they define the degree distribution of a randomly chosen node. In many cases, one is also interested in the node degree associated with a randomly chosen edge. This experiment is biased towards choosing nodes with larger degrees. The resulting degree distributions, $\lambda(x) = L'(x)/L'(1)$ and $\rho(x) = R'(x)/R'(1)$, are called **degree distributions from the edge perspective**. The **design rate** is the code rate implied by dimension of the parity-check matrix and is given by $r = \frac{n-m}{m} = 1 - \frac{P(1)}{\Lambda(1)}$. It equals the true rate of the code if the parity-check matrix is full-rank.

Exercise 3.4. Show that the following relationships hold:

$$L(x) = \frac{\int_0^x \lambda(s) ds}{\int_0^1 \lambda(s) ds} \quad R(x) = \frac{\int_0^x \rho(s) ds}{\int_0^1 \rho(s) ds} \quad r = 1 - \frac{L'(1)}{R'(1)} = 1 - \frac{\int_0^1 \rho(x) dx}{\int_0^1 \lambda(x) dx}.$$

Likewise, the standard ensemble LDPC(n, λ, ρ) of irregular low-density parity-check (LDPC) codes is defined in terms of LDPC(Λ, P) as follows. Let (λ, ρ) an edge-perspective d.d. pair with rational coefficients, (L, R) be the associated node perspective d.d. pair, and $r = 1 - \frac{L'(1)}{R'(1)}$ be the design rate. For any n where nL_i and $n(1-r)R_i$ are integer for all i , one can define $\Lambda(x) = nL(x)$ and $P(x) = n(1-r)R(x)$. For these choices, the ensemble LDPC(n, λ, ρ) is well-defined and equal to LDPC(Λ, P).

Remark 3.5. For values of n where nL_i and $n(1-r)R_i$ are not integer, one option is to define Λ_i, P_i by rounding $nL_i, n(1-r)R_i$. In this case, however, there is no guarantee that $\Lambda'(1) = P'(1)$ (i.e., the number of edges on each side may not match). In practice, this can be fixed up by either reducing the degree of one check node or adding an additional check node.

4 Density Evolution

Definition 4.1. Let $\mathring{C}_\ell(n, \lambda, \rho)$ denote the **ensemble of height- ℓ computation graphs from the node perspective**. A sample \mathbf{T} is drawn from this ensemble by (i) choosing a random element \mathbf{G} from LDPC(n, λ, ρ), (ii) choosing a random variable node v in \mathbf{G} , and (iii) letting \mathbf{T} be the subgraph of \mathbf{G} that induced by all nodes whose distance from v is less than 2ℓ . Likewise, $\vec{C}_\ell(n, \lambda, \rho)$ denotes the **ensemble of height- ℓ computation graphs from the edge perspective**. A sample \mathbf{T} is drawn from this ensemble by (i) choosing a random element \mathbf{G} from LDPC(n, λ, ρ), (ii) choosing a random edge u in \mathbf{G} , and (iii) letting \mathbf{T} be the subgraph of \mathbf{G} induced by the depth- $2\ell + 1$ directed neighborhood of u in the variable node direction.

Example 4.2. Consider the $(\lambda, \rho) = (x, x^2)$ ensemble and observe a length- n code has $\frac{2n}{3}$ check nodes. What is the probability that a randomly chosen element from $\mathring{C}_1(n, x, x^2)$ is a tree? The easiest way to answer this question is by sequentially revealing one edge at a time and computing the probability that the graph remains a tree (i.e., that the revealed edge is “good”). An isolated root node is always a tree. The probability that the code remains a tree after randomly attaching to check nodes at the next level (i.e., after exposing the two edges attached to the root node) is

$$\frac{\#\text{good 1st choices}}{\#\text{1st choices}} \cdot \frac{\#\text{good 2nd choices}}{\#\text{2nd choices}} = \frac{\frac{2n}{3} \cdot 3}{2n} \cdot \frac{(\frac{2n}{3} - 1) \cdot 3}{2n - 1} = \frac{2n - 3}{2n - 1}.$$

During each successful step (where the graph remains a tree), the number of available edges is reduced by 1 and the number of “good” edges is reduced by 3. If the graph remains a tree after a full check node level, then each variable node in the graph either has all edges revealed (i.e., it is in the tree) or no edges revealed (i.e., it is not in the tree). The probability that the code remains a tree after randomly attaching to variable nodes at the next level (i.e., after exposing the four edges attached to the two check nodes at the first level) is

$$\prod_{i=1}^4 \frac{\#\text{good } i\text{-th choices}}{\#\text{}i\text{-th choices}} = \frac{(n-1) \cdot 2}{2n-2} \cdot \frac{(n-2) \cdot 2}{2n-3} \cdot \frac{(n-3) \cdot 2}{2n-4} \cdot \frac{(n-4) \cdot 2}{2n-5} = \prod_{i=1}^4 \frac{2n-2i}{2n-1-i}.$$

During each successful step (where the graph remains a tree), the number of available edges is reduced by 1 and the number of “good” edges is reduced by 2. If the graph remains a tree after a full variable node level, then each check node in the graph either has all edges revealed (i.e., it is in the tree) or no edges revealed (i.e., it is not in the tree).

Exercise 4.3. Enumerate each element of the ensemble $\mathring{C}_1(n, x, x^2)$ along with its probability. Compare your answers with the second column of [1, Fig. 3.37].

Lemma 4.4 (Tree Neighborhoods for LDPC(λ, ρ)). *For any fixed ℓ , the probability that a random element drawn from $\mathring{C}_\ell(n, \lambda, \rho)$ is a tree converges to 1 as $n \rightarrow \infty$.*

Proof. Similar to the example above, one can draw a random sample $T \sim \mathring{C}_\ell(n, \lambda, \rho)$ by first picking the root node and then revealing one new edge connection at a time. The total number of edges in the graph G is $e = nL'(1)$ and the number of edges revealed during the sampling process is a random variable that is upper bounded by $s = 2\ell(\mathbf{r}_{\max}\mathbf{l}_{\max})^\ell$. Observe that each step reduces the number of available edge choices by 1 and the number of “good” edge choices by at most $d = \max(\mathbf{l}_{\max}, \mathbf{r}_{\max})$. In following, we ignore the fact that the number of “good” edge choices resets to the full number of remaining edges after each level is completed. Thus, we have

$$\begin{aligned} \mathbb{P}(T \text{ is a tree}) &\geq \prod_{i=1}^s \frac{\#\text{good } i\text{-th choices}}{\#\text{}i\text{-th choices}} \\ &\geq \prod_{i=1}^s \frac{e - d(i-1)}{e - (i-1)} \geq \left(\frac{e - sd}{e} \right)^s \\ &\geq 1 - \frac{s^2 d}{e} = 1 - \frac{s^2 d}{nL'(1)}. \end{aligned}$$

The stated result follows. □

Remark 4.5. It is easy to verify that the same result holds even if ℓ is allowed to grow logarithmically with n (e.g., $\ell = c \ln n$ for some sufficiently small $c > 0$).

Definition 4.6. Let $\vec{T}_\ell(\lambda, \rho)$ denote the **ensemble of height- ℓ computation trees from the edge perspective**. A sample T is drawn from this ensemble by (i) choosing a variable node degree d according to $\lambda(x)$, (ii) attaching $d - 1$ check nodes whose degrees are drawn i.i.d. according to $\rho(x)$, and (iii) attaching to each open check edge a random element of $\vec{T}_{\ell-1}(\lambda, \rho)$. To start this inductive definition,

$\vec{\mathcal{T}}_0(\lambda, \rho)$ is defined to be the ensemble variable nodes drawn according to $\lambda(x)$. Likewise, $\vec{\mathcal{T}}_\ell(\lambda, \rho)$ denotes the **ensemble of height- ℓ computation trees from the node perspective**. A sample \mathbf{T} is drawn from this ensemble by (i) drawing the root node degree d according the $L(x)$, (ii) attaching d check nodes whose degrees are drawn i.i.d. according to $\rho(x)$, and (iii) attaching to each open check edge a random element of $\vec{\mathcal{T}}_{\ell-1}(\lambda, \rho)$.

Definition 4.7. For any node-perspective (resp. edge-perspective) computation graph or tree \mathbf{T} , let $\mathbb{P}_b^{\text{BP}}(\mathbf{T}, \epsilon)$ the erasure probability of the root node (resp. message on the root edge) given by message-passing decoding when all variables nodes are observed through a BEC(ϵ).

Remark 4.8. The perspective taken by the above ensembles could be described more accurately as considering trees from the variable-node (or node-to-check directed edge) perspective. It is also possible to define tree ensembles from the check-node (or check-to-node directed edge) perspective. For the sake of brevity, we do not define these ensembles explicitly. Though, we may refer to them occasionally in this note.

Theorem 4.9 (DE for LDPC(λ, ρ)). *The average erasure probabilities, of message-passing decoding for height- ℓ tree ensembles, are given by*

$$\begin{aligned} \mathbb{P}_{\vec{\mathcal{T}}_\ell(\lambda, \rho)}^{\text{BP}}(\epsilon) &\triangleq \sum_{\mathbf{T}} \mathbb{P} \left\{ \vec{\mathcal{T}}_\ell(\lambda, \rho) = \mathbf{T} \right\} \mathbb{P}_b^{\text{BP}}(\mathbf{T}, \epsilon) = \epsilon \lambda \left(1 - \rho \left(1 - \mathbb{P}_{\vec{\mathcal{T}}_{\ell-1}(\lambda, \rho)}^{\text{BP}}(\epsilon) \right) \right) \\ \mathbb{P}_{\dot{\mathcal{T}}_\ell(\lambda, \rho)}^{\text{BP}}(\epsilon) &\triangleq \sum_{\mathbf{T}} \mathbb{P} \left\{ \dot{\mathcal{T}}_\ell(\lambda, \rho) = \mathbf{T} \right\} \mathbb{P}_b^{\text{BP}}(\mathbf{T}, \epsilon) = \epsilon L \left(1 - \rho \left(1 - \mathbb{P}_{\vec{\mathcal{T}}_{\ell-1}(\lambda, \rho)}^{\text{BP}}(\epsilon) \right) \right). \end{aligned}$$

Proof. We break the recursive definition of $\vec{\mathcal{T}}_\ell(\lambda, \rho)$ into two steps. Let \mathcal{S} be the half-iteration tree ensemble (i.e., from the check-to-node directed edge perspective) where a sample \mathbf{T}' is generated by drawing a random check degree D according to $\rho(x)$ and then attaching the $D - 1$ open edges to random elements of $\vec{\mathcal{T}}_{\ell-1}(\lambda, \rho)$. The average probability of erasure, under message-passing decoding, for an output edge from ensemble \mathcal{S} is given by

$$\mathbb{P}_{\mathcal{S}}^{\text{BP}}(\epsilon) = \mathbb{E} \left[1 - \left(1 - \mathbb{P}_{\vec{\mathcal{T}}_{\ell-1}(\lambda, \rho)}^{\text{BP}}(\epsilon) \right)^{D-1} \right] = 1 - \sum_{i \geq 1} \rho_i \left(1 - \mathbb{P}_{\vec{\mathcal{T}}_{\ell-1}(\lambda, \rho)}^{\text{BP}}(\epsilon) \right)^{i-1} = 1 - \rho \left(1 - \mathbb{P}_{\vec{\mathcal{T}}_{\ell-1}(\lambda, \rho)}^{\text{BP}}(\epsilon) \right).$$

Let $\vec{\mathcal{T}}_\ell(\lambda, \rho)$ be the full iteration tree ensemble where a sample \mathbf{T} from is generated by drawing a variable node degree D according to $\lambda(x)$ and then attaching the $D - 1$ open edges to random elements of \mathcal{S} . The average probability of erasure for the output message is given by

$$\mathbb{P}_{\vec{\mathcal{T}}_\ell(\lambda, \rho)}^{\text{BP}}(\epsilon) = \epsilon \mathbb{E} \left[\left(\mathbb{P}_{\mathcal{S}}^{\text{BP}}(\epsilon) \right)^{D-1} \right] = \epsilon \sum_{i \geq 1} \lambda_i \left(\mathbb{P}_{\mathcal{S}}^{\text{BP}}(\epsilon) \right)^{i-1} = \epsilon \lambda \left(\mathbb{P}_{\mathcal{S}}^{\text{BP}}(\epsilon) \right).$$

Combining the two half-iteration updates gives the first stated result. The second result uses a similar argument except that the variable degree is drawn according to $L(x)$ and all edge messages are combined. \square

Lemma 4.10 (Monotonicity and Limits of DE). *From Theorem 4.9, one sees that DE update for the standard irregular ensemble is given by $f(\epsilon, x) \triangleq \epsilon \lambda (1 - \rho(1 - x))$. Assuming both $\lambda(x)$ and $\rho(x)$ are not constant, we observe the following:*

1. $f(\epsilon, x)$ is non-decreasing in both arguments for $\epsilon, x \in [0, 1]$ and strictly increasing if $\epsilon, x \in (0, 1)$.
2. For any $x_0, \epsilon \in [0, 1]$, the sequence $x_{\ell+1} = f(\epsilon, x_\ell)$ monotonic in ℓ (e.g., it is increasing if $x_1 > x_0$ and decreasing if $x_1 < x_0$).
3. Let $x_{\ell+1}(\epsilon)$ be defined recursively by $x_{\ell+1}(\epsilon) \triangleq f(\epsilon, x_\ell(\epsilon))$ and $x_0(\epsilon) = 1$. Then, $x_{\ell+1}(\epsilon)$ is non-increasing in ℓ and non-decreasing in ϵ .
4. The function $x_\infty(\epsilon) \triangleq \lim_{\ell \rightarrow \infty} x_\ell(\epsilon)$ exists and is non-decreasing for all $\epsilon \in [0, 1]$.

Proof. For 1., we observe that $\frac{d}{d\epsilon}f(\epsilon, x) = \lambda(1 - \rho(1 - x))$ is positive for $x \in (0, 1]$ and that $\frac{d}{dx}f(\epsilon, x) = \epsilon\lambda(1 - \rho(1 - x))\rho'(1 - x)$ is positive for $x, \epsilon \in (0, 1)$. For 2., the monotonicity of $f(\cdot, \cdot)$ shows that $x_{\ell+1} = f(\epsilon, x_\ell) \geq x_\ell$ implies $x_{\ell+2} = f(\epsilon, x_{\ell+1}) \geq x_{\ell+1}$. Therefore, monotonicity holds inductively and the direction of x_ℓ depends only on the first step. For 3., we first apply 2. with $x_0 = 1$ and observe that $x_1 = \epsilon \leq 1 = x_0$ implies that $x_\ell(\epsilon)$ is non-increasing in ℓ . Next, we observe that $x_0(\epsilon) = 1$ is constant and proceed by induction, for any $\epsilon \leq \epsilon'$, to see that

$$x_{\ell+1}(\epsilon) = f(\epsilon, x_\ell(\epsilon)) \leq f(\epsilon', x_\ell(\epsilon)) \leq f(\epsilon', x_\ell(\epsilon')) = x_{\ell+1}(\epsilon').$$

For 4., the limit exists because 3. implies the sequence $x_\ell(\epsilon)$ is non-increasing in ℓ and bounded for all $\epsilon \in [0, 1]$. The limit function is non-decreasing because 3. implies that, for any $\epsilon \leq \epsilon'$, we have

$$x_\infty(\epsilon) = \lim_{\ell \rightarrow \infty} x_\ell(\epsilon) \leq \lim_{\ell \rightarrow \infty} x_\ell(\epsilon') = x_\infty(\epsilon').$$

This completes the proof. \square

Many important properties of DE follow directly from the monotonicity established in Lemma 4.10. Since $x_\infty(\epsilon)$ is non-decreasing and $x_\infty(0) = 0$, one finds that the DE recursion has a well-defined threshold

$$\epsilon^{\text{BP}}(\lambda, \rho) \triangleq \sup \{ \epsilon \in [0, 1] : x_\infty(\epsilon) = 0 \}.$$

Lemma 4.11. *Let $\epsilon(x) = \frac{x}{\lambda(1-\rho(1-x))}$ be the DE fixed point associated with ϵ . Then, the DE threshold is given by*

$$\epsilon^{\text{BP}}(\lambda, \rho) = \inf_{x \in (0, 1]} \epsilon(x) = \inf_{x \in (0, 1]} \frac{x}{\lambda(1 - \rho(1 - x))}.$$

Proof. If $x \in (0, 1]$ is a fixed point of DE, then we have $x = \epsilon(x)\lambda(1 - \rho(1 - x))$ and one can solve for $\epsilon(x)$. Since convergence requires that $\epsilon^{\text{BP}}(\lambda, \rho)\lambda(1 - \rho(1 - x)) < x$ for $x \in (0, 1]$, one gets an upper bound on $\epsilon^{\text{BP}}(\lambda, \rho)$ for each $x \in (0, 1]$. This implies that

$$\epsilon^{\text{BP}}(\lambda, \rho) \leq \inf_{x \in (0, 1]} \frac{x}{\lambda(1 - \rho(1 - x))}.$$

Conversely, if $\epsilon < \inf_{z \in (0, 1]} \frac{z}{\lambda(1-\rho(1-z))}$, then

$$\epsilon\lambda(1 - \rho(1 - x)) < \inf_{z \in (0, 1]} \frac{z}{\lambda(1 - \rho(1 - z))}\lambda(1 - \rho(1 - x)) \leq \frac{x}{\lambda(1 - \rho(1 - x))}\lambda(1 - \rho(1 - x)) \leq x$$

for $x \in (0, 1]$. This implies that $x_{\ell+1} = \epsilon\lambda(1 - \rho(1 - x_\ell)) < x_\ell$ for $x_\ell \in (0, 1]$ and thus $x_\ell \rightarrow 0$. Therefore, $x_\infty(\epsilon) = 0$ and $\epsilon^{\text{BP}}(\lambda, \rho) \geq \inf_{x \in (0, 1]} \frac{x}{\lambda(1-\rho(1-x))}$. \square

Exercise 4.12. Compute $\text{P}_b^{\text{BP}}(\mathbb{T}, \epsilon)$ for each element of the ensemble $\mathcal{C}_1(n, x, x^2)$. Compare your answers with the third column of [1, Fig. 3.37]. [Beware of double edges... one entry in their table is incorrect.]

5 The Protograph Ensemble

In topology, the idea of a covering map is used to describe large objects that are locally indistinguishable from smaller objects. If the large object can be mapped down to the smaller object in a way that preserves all local properties, then the mapping is called a covering map. Using an appropriate topology, one can apply this idea to graphs. In this section, we will describe how these ideas are used to construct irregular LDPC codes with deterministic local structure. For a more general discussion of structured LDPC codes, see [1, Sec. 7.3].

Definition 5.1. Let B be a **base graph** with no multiple edges and G be an arbitrary graph. A function, $\phi : G \rightarrow B$, that maps the vertices of G to the vertices of B is a **covering map** if, for every $x \in G$, the neighbors of x are bijectively mapped to the neighbors of $\phi(x)$ in B . We say that G is a **graph cover** of B .

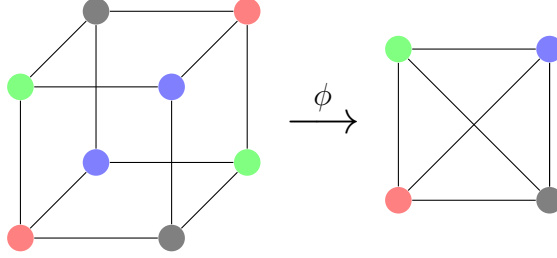


Figure 1: Let ϕ be the map that preserves color in the above diagram. Then, ϕ is a covering map from the cube to the complete graph on 4 vertices.

Example 5.2. Figure 1 shows an example covering map from cube to K_4 , the complete graph on 4 vertices. The function, ϕ , maps each vertex of the cube to the unique vertex in K_4 that has the same color.

The simplest code designs are often based on protographs that have multiple edges (i.e., that are multigraphs). Though the definition above does not generalize cleanly to multigraphs, one can still use them as base graphs in the following construction. Recall that a permutation matrix is a square matrix where each row and column are zero except for a single one.

Definition 5.3. Let B be a bipartite multigraph with $(N - K) \times N$ adjacency matrix H^B . Let H^G be an $m(N - K) \times mN$ matrix formed by replacing the (i, j) entry of H^B with the sum of $H_{i,j}^B$ $m \times m$ permutation matrices. Then, G is an m -lift of B . If B has no multiple edges, then G is also a graph cover of B .

Remark 5.4. The m -lift operation can also be seen as:

1. coloring each edge in the base graph with a different color,
2. copying the colored base graph m times,
3. cutting all edges in half,
4. and, finally, connecting half-edges of the same color in an arbitrary fashion.

Example 5.5. For the base multigraph associated with $H^B = \begin{bmatrix} 3 & 2 \end{bmatrix}$, there is a 3-lift, defined by

$$H^G = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 \end{bmatrix},$$

that has no multiple edges.

Definition 5.6. For a bipartite multigraph B , the **protograph code ensemble** $\text{PROTO}(B, m)$ is defined by choosing the Tanner graph G to be an m -lift of B where each permutation matrix is chosen uniformly at random. The resulting code is defined by the parity-check matrix H^G . For the code ensemble $\text{PROTO}(B, m)$, let $\hat{\mathcal{C}}_\ell(m, B)$ be the **ensemble of height- ℓ computation graphs from the node perspective**.

Remark 5.7. Let G be an m -lift of the base graph B . Then, the copy-and-permute construction implies a natural projection $\phi : G \rightarrow B$ defined by mapping $x \in G$ to the node in B that was copied to generate x . If the base graph does not have multiple edges, then this mapping is a covering map.

Remark 5.8. For the protograph ensemble $\text{PROTO}(B, m)$, each row and column in the parity-check matrix H^B of the base graph B defines a **node type**. The local neighborhood of each node in the lifted graph is deterministic and depends only on its node type (i.e., its associated node in the base graph). Likewise, each non-zero entry in the parity-check matrix H^B of the base graph defines an **edge type**. The local neighborhood of each edge in the lifted graph is deterministic and depends only on its edge type.

Lemma 5.9. For any fixed ℓ , the probability that a random element drawn from $\hat{\mathcal{C}}_\ell(m, B)$ is a tree converges to 1 as $m \rightarrow \infty$.

Proof. The code ensemble $\text{PROTO}(B, m)$ is defined by choosing one random permutation for each edge in B . Similar to the LDPC(n, λ, ρ) ensemble, one can draw a random sample $\mathbf{T} \sim \hat{\mathcal{C}}_\ell(n, B)$ by first picking the root node and then revealing one unknown element of a permutation at each step. The deterministic local neighborhoods of the lifted graph imply that, given the node type of the root node, there is exactly one computation tree of height ℓ . Let s be the number of edges in that tree and $d = \max_{i,j} H_{i,j}^B$ be the maximum edge multiplicity in the base graph. First, we observe that the number of ways to attach an edge to a socket is always upper bounded by m . Next, we notice that revealing an edge reduces the number of “good” choices in any other permutation by at most d . Thus, we have

$$\begin{aligned} \mathbb{P}(\mathbf{T} \text{ is a tree}) &\geq \prod_{i=1}^s \frac{\#\text{good } i\text{-th choices}}{\#\text{}i\text{-th choices}} \\ &\geq \prod_{i=1}^s \frac{m - d(i-1)}{m} \geq \left(\frac{m - sd}{m}\right)^s \\ &\geq 1 - \frac{s^2 d}{m}. \end{aligned}$$

The stated result follows. \square

Theorem 5.10. Let $\mathbf{T}_{ij}^{(\ell)}$ be the edge-perspective computation tree associated with the (i, j) entry of H^B after ℓ iterations of decoding and define $x_{i,j}^{(\ell)} = \mathbb{P}_b^{\text{BP}}(\mathbf{T}_{ij}^{(\ell)}, \epsilon)$. Then, the deterministic neighborhoods of each edge imply the recursion

$$x_{i,j}^{(\ell)} = \epsilon \prod_{i'=1}^{N-K} \left(1 - \prod_{j'=1}^N \left(1 - x_{i',j'}^{(\ell-1)} \right)^{H_{i',j'}^B - \delta_{j,j'}} \right)^{H_{i',j}^B - \delta_{i,i'}}.$$

Proof. Let $y_{i',j}^{(\ell)}$ be the erasure probability of check-to-variable messages for edge-type (i', j) after ℓ iterations. Since the check node associated with row i' of H^B has exactly $H_{i',j'}^B$ edges of type (i', j') , one finds that

$$y_{i',j}^{(\ell)} = 1 - \prod_{j'=1}^N \left(1 - x_{i',j'}^{(\ell-1)} \right)^{H_{i',j'}^B - \delta_{j,j'}},$$

where the Kronecker delta function $\delta_{j,j'}$ is used to implement the “leave-one-out” message-passing rule. Since the variable node associated with column j of H^B has exactly $H_{i',j}^B$ edges of types (i', j) , one finds that

$$x_{i,j}^{(\ell)} = \epsilon \prod_{i'=1}^{N-K} \left(y_{i',j}^{(\ell)} \right)^{H_{i',j}^B - \delta_{i,i'}},$$

where $\delta_{i,i'}$ is again used to implement the “leave-one-out” rule. The stated result follows from combining these two expressions. \square

Exercise 5.11. The monotonicity of protograph DE can be established by defining the partial order $X \preceq X'$ to mean $[X]_{i,j} \leq [X']_{i,j}$ for all i, j . Using this partial order, try to generalize each part of Lemma 4.10 to protograph DE.

Exercise 5.12. Use a computer (e.g., write a program) to numerically compute the rate and DE threshold of the protograph ensembles defined by

$$H^B = \begin{bmatrix} 3 & 3 \end{bmatrix}, \quad H^B = \begin{bmatrix} 3 & 2 \end{bmatrix}, \quad H^B = \begin{bmatrix} 4 & 2 \end{bmatrix}, \quad H^B = \begin{bmatrix} 4 & 4 \end{bmatrix}, \quad H^B = \begin{bmatrix} 2 & 2 & 2 & 1 \\ 2 & 1 & 0 & 1 \end{bmatrix}.$$

