Abstract—Product codes (that are deterministic constructions) and Low-Density-Parity-Check (LDPC) codes (that come from a random ensemble of codes on graphs) have had little overlap in design and analysis methodology. We show that when nature offers suitable randomness in the form of the underlying statistical communications channel (e.g. discrete symmetric erasure and error channels), these two families of codes can be unified via their pruned residual Tanner graph post channel-corruption, allowing product codes to cross-leverage the design and analysis literature of LDPC codes. Further, for a subclass of these product codes, we can leverage the power of density evolution methods used successfully for LDPC codes. In this work, focusing on symmetric erasure channels, we show that \(d\)-dimensional product codes are isomorphic to a class of \(d\)-left regular Generalized Low-Density-Parity-Check (GLDPC) codes. This insight allows us to characterize the performance of \(d\)-dimensional product codes with component codes having constant erasure correcting capability, with relative simplicity using density evolution methods. The isomorphism also reveals that the choice of 3D product codes with a single parity-check per component offers superior performance at asymptotically high rates for the erasure channel to the predominantly-studied 2D product codes. Indeed, 3-dimensional product codes having a single parity check per component perform identically to 3-left-regular LDPC codes, which for the erasure channel (using density evolution methods) are about 22\% away from Shannon capacity. This, combined with the simplicity and linear-time encoding and decoding of product codes, makes them attractive to a host of applications, with one particularly exciting one being their key role in performing coded distributed matrix multiplication \(\Box\) for straggler mitigation in modern large-scale distributed computing platforms.

I. INTRODUCTION

Product Codes and Low-Density-Parity-Check (LDPC) codes have co-existed separately for decades with little effort to unify them in design and analysis. This may have been in part due to product codes being deterministic constructions whereas LDPC codes come from a random ensemble of codes on graphs. However, when nature offers suitable randomness in the form of the underlying statistical communications channel, these two families of codes can be unified via their pruned residual graph post channel-corruption. Important examples include discrete symmetric erasure and error channels, and more generally, discrete channels that render every subset of encoded symbols equally likely to be corrupted. We show how for these settings it is possible to unify these two disparate families of codes, allowing product codes to cross-leverage the design and analysis literature of LDPC codes. We further show that for a subclass of these product codes, we can leverage the power of density evolution methods used successfully for LDPC codes.

Specifically, focusing on symmetric erasure channels in this work, we show that \(d\)-dimensional product codes are isomorphic to a class of \(d\)-left regular Generalized Low-Density-Parity-Check (GLDPC) codes. This insight allows us to characterize the performance of \(d\)-dimensional product codes with component codes having constant erasure correcting capability, with relative simplicity using density evolution methods in contrast to the more cumbersome analytical tools typically deployed in this literature. The isomorphism also reveals that while the bulk of the product code literature has focused on 2D codes with “heavy” component codes, the choice of 3D codes with a single parity-check per component offers superior performance at asymptotically high rates for the erasure channel\(\Box\)

While the rigorous proof is more detailed, the underlying idea is quite simple; the way we are able to draw such an isomorphism between deterministic and random constructions is by utilizing the random appearance of the pruned residual graph. Before being sent through the channel, a \(d\)-dimensional product code has a deterministic Tanner graph; there is no ensemble of graphs to analyze. However, the channel randomly corrupting a subset of symbols allows us to analyze the residual graph with only those corrupted symbols as variable nodes. Since a random subset is corrupted, the residual graph appears sufficiently random, and so we are able to analyze the ensemble of pruned residual graphs.

A. Product Codes

Product codes have a very intuitive and regular structure that makes them easy to conceptualize, implement, and understand. In standard product codes, \(d = 2\), one selects two systematic linear codes, \(C_1 = (n_1, k_1, d_1)\) and \(C_2 = (n_2, k_2, d_2)\). The

1While \(d\)-dimensional product codes are asymptotically rate 1, the term is misleading; while using the traditional definition of rate, a code of length \(N\) with \(k = N - N^\delta\) is rate 1, these constructions aren’t simply a point on the spectrum. For any fixed values of \(N, K\), one can find a \(\delta\) such that \(K = N - N^\delta\). Practically, these asymptotically rate 1 schemes translate to around 10\% redundancy, which is very practical in many applications.
resulting code is \((n_1n_2, k_1k_2, d_1d_2)\), which is achieved by arranging the data bits in a \(k_2 \times k_1\) rectangle, and encoding to a \(n_2 \times n_1\) rectangle. This is done by encoding the \(k_2\) data rows with \(C_1\), determining the last \(n_1 - k_1\) bits of each row, and then encoding all \(n_1\) columns with \(C_2\) to determine the last \(n_2 - k_2\) bits of each column. Since \(C_2\) is linear, the last \(n_2 - k_2\) rows are also codewords of \(C_1\). These product codes are very appealing due to their deterministic construction and linear encoding time. This construction also allows for the use of a peeling decoder, which yields simple, linear time decoding.

\(d\)-Dimensional product codes are a natural extension of this 2-dimensional framework. For the sake of clarity, we focus on the case where the component codes in each dimension are the same, \(C = (n, k, d_{\text{min}})\), but our analysis can be extended to the case where different component codes are used in each dimension. We additionally assume that \(C\) is a linear code and has been put in systematic form. To obtain our product code, we form the data into a \(d\)-dimensional hypercube with side length \(k\) and index each data location with its Cartesian coordinates, \((l_1, l_2, ..., l_d)\), with \(l_i = 0, ..., k - 1\). We encode this to a \(d\)-dimensional hypercube with side length \(n\) in a similar manner to the 2D case, such that \(\forall i \in \{1, 2, ..., d\} : l_i = 0, ..., n - 1\) \(\in C\). This means that every “row” is a codeword, with a \(d\)-dimensional row being a set of \(n\) elements obtained by fixing all but 1 index, and varying that 1 index \(0, ..., n - 1\).

### B. LDPC Background

Low Density Parity Check (LDPC) codes were first introduced by Gallager in 1962 [2]. As they’ve evolved, more complex variants have been proposed, like Generalized LDPC codes (GLDPC). A GLDPC code is a low-density parity check code in which the constraint nodes of the code graph are arbitrary linear block codes, rather than just single parity checks [3]. LDPCs are very appealing in their ease of analysis, particularly using density evolution.

### III. MAIN RESULT

We provide a new lens through which to view \(d\)-dimensional product codes, and provide a rigorous asymptotic analysis of their performance over the erasure channel using density evolution. We bridge the gap between the random construction of LDPC codes and the deterministic construction of product codes by using the randomness of the erasure channel in generating the residual graph.

A similar analysis can be done for other discrete symmetric channels, like the q-ary symmetric channel; the isomorphism will still hold, as there exists a random subset of corrupted variable nodes, and so the residual appears to have been generated randomly. Instead of the simple density evolution equation, however, a more complicated message passing algorithm will need to be analyzed. For readability, we restrict our focus to the erasure channel for the rest of the paper.

### A. Code Construction

In this paper we present two constructions for \(d\)-dimensional product codes. The first we dub a “Pseudo Product Code”, which entails a single parity check code applied to every \(d - 1\) dimensional slice (this can be generalized to a systematic, linear \((n^{d-1}, k, d)\) code). An example for a 3D Pseudo product code can be found in Fig. 1(a) with the colored lines representing the parities for each plane (2 dimensional slice). This construction is similar to [4], but where our \(d\)-dimensional pseudo product code has \(n(d - 1) + 1\) parity bits (as one is a global parity, common to all stages), their alternate construction has \(nd\) parities added to the outside of the hypercube of data, instead of woven into it.

A standard Product Code, can be viewed as having a parity check (or arbitrary systematic linear \((n, k, d)\) component code) on every 1 dimensional slice (row). This translates to Fig. 1(b) where each colored plane represents the generalized parity checks in a given dimension.
Lemma 1: A d-dimensional product code with a systematic linear $C = (n,k,d_{\text{min}})$ component code is equivalent to the Tanner graph underlying a deterministically constructed $d$-left regular GLDPC code with $C$ at each check node.

Proof: We note that for a product code, in each dimension there are $n^{d-1}$ rows, each representing a constraint on the length $N$ codeword. In order to construct an equivalent GLDPC code, we simply need to have $d_n, d_{\text{min}}$ check nodes arranged in $d$ stages of $n^{d-1}$ check nodes each (one representing each “face” of the $d$-dimensional hypercube). We take the $N$ code indices, and for each stage map them to the check node corresponding to their row in that particular stage. The check nodes (colored planes) will apply the constraint implied by the $C$ matrix of the length $n$ codeword there are $n$ check nodes at each check node.

A $d$-dimensional product code with a linear component code is linear as well, and can thus be characterized by its generator matrix. Fig. 2 shows such an example generator matrix.

$$G = \begin{pmatrix}
1 & 1 & \ldots & 1 \\
1 & 1 & \ldots & 1 \\
\vdots & \vdots & \ddots & \vdots \\
1 & 1 & \ldots & 1 \\
\end{pmatrix}$$

Fig. 2. Generator Matrix of $d$-dimensional product code with $C = (4,3,2)$

For any linear component code $C$, the generator matrix of the $d$-dimensional product code corresponding to $C$ can be expressed as the tensor product of $d$ copies of the generator matrix of $C$.

IV. PSEUDO PRODUCT CODE

We now show that a class of deterministically constructed GLDPC codes have residual graphs equivalent to those of a traditional balls and bins model, after transmission through the BEC.

A. Ensemble $C_0^d(n,d)$ of bi-partite graphs constructed using a balls and bins model

Bipartite graphs in the ensemble $C_0^d(n,d)$ have $\ell$ variable nodes on the left, and $dn$ check nodes on the right. We group these check nodes into $d$ stages, each containing $n$ check nodes. Each variable (left) node is connected to $d$ check nodes, which are selected uniformly at random, 1 from each stage of check nodes.

B. Ensemble $C_1^d(n,d,\Phi)$ of bipartite graphs constructed using the mapping $\Phi$

For a given mapping $\Phi : \mathbb{Z}_N \mapsto \mathbb{Z}_n \times \ldots \times \mathbb{Z}_n$, $\Phi(v) = (\Phi_1(v), \ldots, \Phi_d(v))$, the ensemble $C_1^d(n,d,\Phi)$ of $d$ left regular bipartite graphs with $\ell$ variable nodes, and $dn$ check nodes, is defined as follows. As before, we partition the check nodes into $d$ sets of $n$ check nodes each. Now, we consider a set $\mathcal{I}$ of $\ell$ integers, where each element of the set $\mathcal{I}$ is between 0 and $N-1$. Assign the $\ell$ integers from the set $\mathcal{I}$ to the $\ell$ variable nodes in arbitrary order. Label the check nodes in each set from 0 to $n-1$, for all $i = 1, \ldots, d$. A $d$-left regular bipartite graph with $\ell$ variable nodes and $dn$ check nodes is then obtained by connecting a variable node with an associated integer $v$ to check node $\Phi_i(v)$ in stage $i$, for $i = 1, \ldots, d$. The ensemble $C_1^d(n,d,\Phi)$ is the collection of $d$-left regular bipartite graphs induced by all possible sets $\mathcal{I}$.

Lemma 2: For any isomorphism $\Phi$, the ensembles $C_1^d(n,d)$ and $C_2^d(n,d,\Phi)$ are identical.

Proof: It is trivial to see that $C_1^d(n,d,\Phi) \subset C_2^d(n,d)$. Now, we show the reverse. Consider a graph $G_1 \in C_1^d(n,d)$. Suppose a variable node $v \in G_1$ is connected to the check nodes numbered $\{f_i\}_{i=1}^{d-1}$. Since $\Phi$ is an isomorphism, we know there exists $\Phi^{-1} : \mathbb{Z}_n \mapsto \mathbb{Z}_n$, and thus we can find an integer $q = \Phi^{-1}(f_i)$ between 0 and $n^d-1$ such that $\Phi_i(q) = f_i \forall i = 1, \ldots, d$. Thus, for every graph $G_1 \in C_1^d(n,d)$, there exists a set $\mathcal{I}$ of $k$ integers that will result in an identical graph using the $\Phi$ based construction, and so $G_1 \in C_2^d(n,d,\Phi)$. Hence, $C_1^d(n,d) = C_2^d(n,d,\Phi)$.

Using the cartesian mapping for $\Phi$, we are able to obtain a pseudo product code by mapping to $\Phi$ as in $C_0^d(n,d,\Phi)$ for $\Phi(l) = (l/n^{d-1}, \ldots, (l/n^{d-1})%n, \ldots, l%n)$.

V. STANDARD PRODUCT CODE

Now that we have established the above, we continue to a traditional $d$-dimensional product code. We can see an example 3D product code in Fig. 3. We can analyze the performance of this scheme in a similar manner to the Pseudo Product Code.

A. Balls-and-Bins Construction

We construct a bipartite graph with $\ell$ variable nodes on the left, and $dn^{d-1}$ check nodes on the right as follows. We partition the check nodes into $d$ stages of $n^{d-1}$ check nodes each, and arrange them in a $d-1$ dimensional hypercube with
side length $n$. We index the check nodes for a given stage by a length $d-1$ tuple, with indices ranging from $0 \to n-1$, representing its coordinates (like the cartesian coordinates above). Each variable node selects a tuple uniformly at random from $\mathbb{Z}_n^d = (v_1, v_2, \ldots, v_d)$. These tuples are chosen uniformly at random and with replacement for all $\ell$ variable nodes. For each variable node $(v_1, v_2, \ldots, v_d)$, we connect it to the check node indexed by $(v_1, v_2, \ldots, v_{i-1}, v_{i+1}, \ldots, v_d)$ in stage $i$ for $i = 1, \ldots, d$. We refer to this ensemble as $C_\ell^d(n,d)$.

B. $\Phi$ Guided Construction

For any $\Phi : \mathbb{Z}_n \mapsto \mathbb{Z}_n \times \cdots \times \mathbb{Z}_n$ with $\Phi(v) = (\Phi_1(v), \ldots, \Phi_d(v))$, we generate graphs in the following manner: choose $\ell$ integers at random, with replacement, from $0, \ldots, N-1$. Call this set $\mathcal{I}$. Partition the $dn^{d-1}$ check nodes into $d$ stages as before, arranged and indexed in a $d-1$ dimensional hypercube with side length $n$. For convenience, we define $\Phi'_j(v) \triangleq (\Phi_1(v), \ldots, \Phi_{j-1}(v), \Phi_{j+1}(v), \ldots, \Phi_d(v))$.

For any $j \in \mathcal{I}$, we attach node $v$ to check node $\Phi'_j(v)$ in stage $j$, for $j = 1, \ldots, d$. We refer to this ensemble as $C^d_\ell(n,d,\Phi)$.

Lemma 3: For any isomorphism $\Phi$, the ensembles $C^d_\ell(n,d)$ and $C^d_\ell(n,d,\Phi)$ are equivalent.

Proof: The proof is identical to Lemma 2.

C. Implication

Theorem 1: For any isomorphism $\Phi : \mathbb{Z}_n \mapsto \mathbb{Z}_n \times \cdots \times \mathbb{Z}_n$, the deterministically constructed GLDPC code constructed by mapping each variable node $v = 0, \ldots, n^d-1$ to check nodes $\Phi'(v)$ in stages $1, \ldots, d$ as in $C^d_\ell(n,d,\Phi)$ can be analyzed using density evolution. This also holds using the mapping $\Phi$ as in $C^d_\ell(n,d,\Phi)$.

Proof: For any isomorphism $\Phi : \mathbb{Z}_n \mapsto \mathbb{Z}_n \times \cdots \times \mathbb{Z}_n$, picking a residual graph at random from $C^d_\ell(n,d,\Phi)$ is equivalent to picking a graph at random from the ensemble $C^d_\ell(n,d)$ generated using the balls and bins model, as the two ensembles are equivalent by Lemma 3. Similarly, by Lemma 2, $C^d_\ell(n,d) = C^d_\ell(n,d,\Phi)$. This means we can use density evolution to analyze graphs deterministically generated by $\Phi$ and randomly erased by nature.

Again, we use the cartesian mapping for $\Phi$, and are able to obtain a standard product code by mapping to $\Phi'$, as in $C^d_\ell(n,d,\Phi')$, using same $\Phi(l) = (l/n^{d-1}, \ldots, l/n^{d-1}/n, \ldots, l/n)$. We refer to this as $C^d_\ell(n,d,\Phi')$.

VI. PERFORMANCE ANALYSIS

We proceed to analyze the asymptotic performance of such a $d$-dimensional product code. We restrict our performance analysis to $(n,k,r+1)$ component codes with constant $r$.

Theorem 2: $d$-dimensional product codes with $(n,k,r+1)$ component codes with linear encoding and decoding time have the following properties:

1) Can asymptotically correct for $\eta(n,r)$, $n$ erasures
2) Can be encoded and decoded in linear time

For constants $1.22 < \eta(d,r) < 2$ for most useful regimes, Table 2

Proof: The ensemble of residual graphs of these $d$-dimensional product codes is equivalent to $C^d_\ell(n,d,\Phi)$ by Lemma 2 which by Theorem 1 is equivalent to the ensemble of residual graphs generated with the balls-and-bins model. Thus, we draw conclusions about the performance of product codes by analyzing the ensemble of balls and bins residual graphs. There are three main steps to analyzing the performance of this ensemble of balls-and-bins graphs, which we will briefly outline in the following subsections. More detailed proofs can be found in [9].

A. Density Evolution

We analyze a message-passing algorithm’s performance over a typical graph from the ensemble. We assume the graph has local neighborhoods that are cycle free up to depth $2l$. Thus, for the first $l$ rounds of message passing, messages are independent.

B. Convergence to Cycle-free Case

We can show that the expected behavior of graphs in the ensemble $C^d_\ell(n,d)$ converges to cycle-free. Furthermore, the proportion of edges left undecoded after $l$ iterations is tightly concentrated around $p_l$, the proportion density evolution tells us should be undecoded.

C. Expander Graph

We show that graphs in the ensemble $C^d_\ell(n,d)$ are expanders with high probability. This means that if our peeling decoder decodes all but a small fraction of variable nodes, then it will decode all the variable nodes.

For $d=2$, we cannot prove this last portion. In fact, we can show that the ensemble of graphs $C^1_\ell(n,2)$ are not expanders graphs with probability bounded away from zero. We can see that our peeling decoder will not be able to resolve a graph $G_1 \in C^1_\ell(n,2)$ if there are two variable nodes in $G_1$ that share all the same check nodes. We examine this scenario by relating it to the birthday paradox. If $l$ people have birthdays randomly selected between $0, \ldots, n^d-1$, then the probability that at least two people have the same birthday is approximately $1 - e^{-\ell^2/2n^2}$. In our scenario, the number of erased variable nodes is $\ell = O(n^{d-1})$, and so for $d \leq 2$, the probability that two people have the same birthday, i.e. share all the same check nodes, is bounded away from zero. This means that the expander graph condition is not met, yielding an error floor.

VII. SIMULATION RESULTS

A. Thresholds

Now that we have established an isomorphism between $d$-dimensional product codes and $d$-left regular GLDPC codes, we can analyze the asymptotic performance of $d$-dimensional product codes through density evolution analysis on $d$-left regular GLDPC codes.

2We note that there is a difference between drawing the set $\mathcal{I}$ with and without replacement, but this difference does not affect the analysis as in Proposition 5 in [10].
We use the accepted notation for the edge-degree distribution polynomials of the bipartite graphs in the ensemble as \(\lambda(\alpha) \equiv \sum_{i=1}^{\infty} \lambda_i \alpha^{i-1}\) and \(\rho(\alpha) \equiv \sum_{i=1}^{\infty} \rho_i \alpha^{i-1}\) where \(\lambda_i\) (resp. \(\rho_i\)) denotes the probability that an edge of the graph is connected to a left (resp. right) node of degree \(i\). Thus for the residual graphs of a \(d\)-dimensional product code, equivalently the ensemble \(\mathcal{C}_d^r(n, d)\) constructed using the balls-and-bins procedure, \(\lambda(\alpha) = \alpha^{-d}\), as the graph is \(d\)-left regular. The edge degree distribution \(\rho(\alpha)\) can be derived as in [2] to be \(\rho(\alpha) = \exp(-d(1-\alpha) / \eta(d, r))\). Under the tree-like (cycle-free) assumption that we make, we can write an equation for \(p_j\), the probability that an edge in the Tanner graph is left undecoded after \(j\) iterations, starting with \(p_0 = 1\).

\[
p_{j+1} = \lambda \left(1 - \sum_{i=0}^{r-1} (1 - p_i)\right)
\]

We can specialize this density evolution equation to the case of \(\mathcal{C}_d^r(n, d)\), obtaining the update equation [1]

\[
p_{j+1} = \left(1 - \sum_{i=0}^{r-1} \frac{dp_i}{d(\eta(d, r))} \rho^{-d-1} \eta d i! \right) d^{-1}
\]

This density evolution process [1] will converge to 0 in \(\ell\) iterations for sufficiently large \(\eta(d, r)\). We list minimum values of \(\eta(d, r)\) for various \((d, r)\) in Table I.

### Table I

<table>
<thead>
<tr>
<th>(d)</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>1.2218</td>
<td>1.2580</td>
<td>1.3797</td>
<td>1.4564</td>
<td>1.5202</td>
<td>1.5741</td>
<td>1.6212</td>
</tr>
<tr>
<td>4</td>
<td>1.2949</td>
<td>1.4998</td>
<td>1.6568</td>
<td>1.7781</td>
<td>1.8760</td>
<td>1.9575</td>
<td>2.0275</td>
</tr>
<tr>
<td>5</td>
<td>1.4250</td>
<td>1.7275</td>
<td>1.9409</td>
<td>2.1031</td>
<td>2.2327</td>
<td>2.3406</td>
<td>2.4329</td>
</tr>
<tr>
<td>6</td>
<td>1.5697</td>
<td>1.9577</td>
<td>2.2244</td>
<td>2.4256</td>
<td>2.5864</td>
<td>2.7199</td>
<td>2.8341</td>
</tr>
<tr>
<td>7</td>
<td>1.7189</td>
<td>2.1869</td>
<td>2.5051</td>
<td>2.7446</td>
<td>2.9361</td>
<td>3.0953</td>
<td>3.2309</td>
</tr>
</tbody>
</table>

Asymptotically, a sharp phase transition can be observed in \(\eta(d, r)\), and thus the code will be able to correct for \(n^2 - k^2\) erasures, for component code \((n, k, r+1)\). We can see from density evolution fixed point analysis that out of all \(d \geq 3, r \geq 1\), it is optimal for a given rate to choose \(d=3, r=1\). We ignore \(d=2\) due to the inherent error floor of \(\mathcal{C}_d^r(n, d)\), as explained with the birthday paradox.

### VIII. Conclusions and Future Work

We have shown that when nature supplies suitable randomness in the form of the communication channel, product codes and LDPC codes can be unified in their design and analysis via their pruned residual Tanner graph, allowing product codes to leverage the design and analysis tools of LDPC codes.

Concretely, we have shown that under a discrete symmetric channel, the ensemble of residual graphs generated by transmitting deterministically constructed product codes is isomorphic to the ensemble of residual graphs generated by transmitting randomly constructed GLDPC codes using a balls and bins model. We have further shown that the subset of these with component codes that correct for a constant number of erasures can be analyzed using the powerful and elegant method of Density Evolution. Our isomorphism provides new insights into the design and performance analysis of \(d\)-dimensional product codes, and unveils the superiority of 3D product codes over the well-studied class of 2D product codes for the erasure channel at high rates. While we have limited our isomorphism to the equivalence of the ensemble of pruned Tanner graphs in this work, we conjecture that this isomorphism can be extended to more general settings as well, which will be part of our future work. Further, while we have limited our density evolution analysis of \(d\)-dimensional product codes at high rates, we believe that this can be extended to more general rate settings as well. We are excited about the potential for further insights, and will explore this as part of future work.

### References


