The Distribution of Cycle Lengths in Graphical Models for Turbo Decoding

Technical Report UCI-ICS 99-10
Department of Information and Computer Science
University of California, Irvine

Xianping Ge, David Eppstein, Padhraic Smyth
Information and Computer Science
University of California, Irvine
{xge,eppstein,smyth}@ics.uci.edu

March, 1999

Abstract

This paper analyzes the distribution of cycle lengths in turbo-decoding graphs. The properties of such cycles are of significant interest since it is well-known that iterative decoding can only be proven to converge to the correct posterior bit probabilities (and bit decisions) for graphs without cycles. We estimate the probability that there exist no simple cycles of length less than or equal to k at a randomly chosen node in the graph using a combination of counting arguments and independence assumptions. Simulation results validate the accuracy of the various approximations. For example, for a block length of 64000 a randomly chosen node has a less than 1% chance of being on a cycle of length less than or equal to 10, but has a greater than 99.9% chance of being on a cycle of length less than or equal to 20. The effect of the "S-random" permutation is also analyzed and it is shown that while it eliminates short cycles of length $k < 8$, it does not significantly affect the overall distribution of cycle lengths. The paper concludes by commenting briefly on how these results may provide insight into the practical success of iterative decoding methods.

1 Introduction

Turbo codes are a new class of error-control coding systems that offer near optimal performance while requiring only moderate complexity [1]. It is now known that the widely-used iterative decoding algorithm for turbo-codes is in fact a special case of a general local message-passing algorithm for efficiently computing posterior probabilities in acyclic directed graphical (ADG) models (also known as "belief networks") [2][3]. Thus, it is appropriate to analyze the properties of iterative-decoding by analyzing the properties of the associated ADG model.

In this paper we derive analytic approximations for the probability that a randomly chosen node in the graph for a turbo-code participates in a simple cycle of length less than or equal to k. The resulting expressions provide insight into the distribution of cycle lengths in turbo-decoding For example, for block lengths of 64000, a randomly chosen node in the graph participates in cycles of length less than or equal to 8 with probability 0.002, but participates in cycles of length less than or equal to 20 with probability 0.9998.

In Section 2 we review briefly the idea of ADG models, define the notion of a *turbo graph* and the related concept of a *picture*, and discuss how the cycle-counting problem can be addressed by analyzing how pictures can be embedded in a turbo graph. With these basic tools we proceed in Section 3 to obtain closed-form expressions for the number of pictures of different lengths and obeying different constraints. In Section 4 we derive upper and lower bounds on the probability of embedding a picture in a turbo-graph at a randomly chosen node. Using these results, in Section 5 we derive approximate expressions for the probability of no simple cycles of length k or less. Section 6 shows that the derived analytical expressions are in close agreement with simulation. In Section 7 we investigate the effect of the S-random permuter construction. Section 9 contains a discussion of what these results may imply for iterative decoding in a general context.

2 Background and Notation

2.1 Graphical Models for Turbo-codes

An ADG model (also known as a belief networks) consists of a both a directed graph and an associated probability distribution over a set of random variables of interest. (Note that ADG is the more widely used terminology in the statistical literature, whereas the term belief network or Bayes network is more widely used in computer science; however, both frameworks are identical). There is a 1-1 mapping between the nodes in the graph and the random variables. Loosely speaking, the presence of a directed edge from node A to B in the graph means that B is assumed to have a direct dependence on B (" A causes B"). More generally, if we identify $\pi(A)$ as the set of all *parents* of A in the graph (namely, nodes which point to A), then A is conditionally independent of all other nodes in the graph (except for A's descendants) given the values of the variables in the set $\pi(A)$. The familiar Markov chain is a special case of such a graph, where each variable has a single parent. The general ADG model framework is quite powerful in that it allows us to model and analyze independence relations among relatively large and complex sets of random variables.

As originally shown in [2, 3, 4], turbo-codes can be usefully cast in an ADG framework. Figure 1 shows the ADG model for a rate 1/2 turbo code. The **U** nodes are the original information bits to

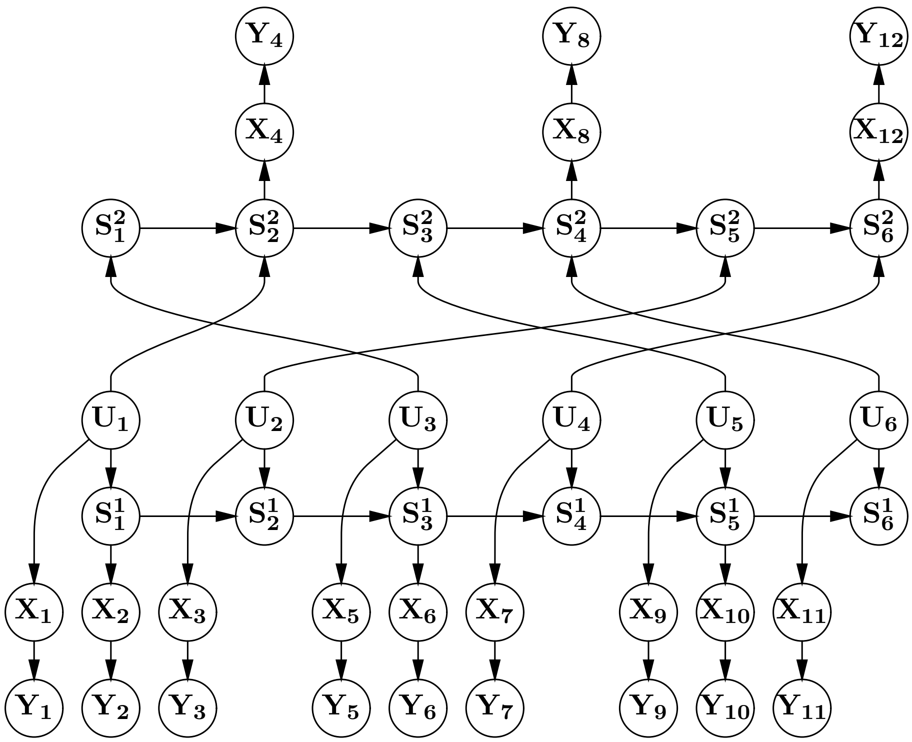


Figure 1: The ADG model for a $K = 6, N = 12$, rate 1/2 turbocode.

be coded, the **S** nodes are the linear feedback shift register outputs, the **X** nodes are the codeword which is the input to the communication channel, and the **Y** nodes are the channel outputs. The ADG model captures precisely the conditional independence relations which are implicitly assumed in the turbo-coding framework, e.g., the input bits **U** are marginally independent, the state nodes **S** only depend on the previous state and the current input bit, and so forth.

The second component of an ADG model (in addition to the graph structure) is the specification of a joint probability distribution on the random variables. A fundamental aspect of ADG models is the fact that this joint probability distribution decomposes into a simple factored form. Letting $\{A_1, \ldots, A_n\}$ be the variables of interest, we have

$$p(A_1, \ldots, A_n) = \prod_{i=1}^{n} p\left(A_i | \pi(A_i)\right), \tag{1}$$

i.e., the overall joint distribution is the product of the conditional distributions of each variable A_i given its parents $\pi(A_i)$. (We implicitly assume discrete-valued variables here and refer to distributions; however, we can do the same factorization with density models for real-valued variables, or with mixtures of densities and distributions).

To specify the full joint distribution, it is sufficient to specify the individual conditional distributions. Thus, if the graph is sparse (few parents) there can be considerable savings in parameterization of the model. From a decoding viewpoint, however, the fundamental advantage of the factorization of the joint distribution is that it permits the efficient calculation of posterior probabilities of interest, i.e., if the values for a subset of variables are known (e.g., the received codewords **Y**) we can efficiently compute the posterior probability for $\mathbf{U_i} = 1$, $1 \leq i \leq N$. The power of the ADG framework is that there exist exact local message-passing algorithms which calculate such posterior probabilities. These algorithms have time complexity which is linear in the diameter of the underlying graph times a factor which is exponential in the cardinality of the variables at the nodes in the graph. The algorithm is provably convergent to the true posterior probabilities provided the graph structure does not contain any loops. The message-passing algorithm of Pearl [5] is perhaps the best-known of such algorithms. For regular convolutional codes, Pearl's message passing algorithm applied to the convolutional code graph structure (e.g., the lower half of Figure 1) yields the BCJR decoding algorithm [6].

If the graph has loops, such as the turbo-code structure of Figure 1, Pearl's algorithm no longer provably converges. In essence, the messages being passed can arrive at the same node via multiple paths, leading to multiple "over-counting" of the same information. A widely used strategy in statistics and artificial intelligence is to reduce the original graph with loops to an equivalent graph without loops (this can be achieved by clustering variables in a judicious manner) and then applying Pearl's algorithm to the new graph. However, if we apply this method to ADGs for realistic turbo-codes the resulting "no loop" graph will contain at least one node with a large number of variables. Consequently, this node will have cardinality exponential in this number, leading to exponential complexity in the probability calculations referred to above (in the limit all variables are combined into a single node and there is no factorization in effect). Thus, for turbo-codes, there is no known efficient exact algorithm for computing posterior probabilities (i.e., for decoding).

However, as shown in [2, 3, 4], the iterative decoding algorithm of [1] can be shown to be equivalent to applying the local-message passing algorithm of Pearl directly to the ADG structure

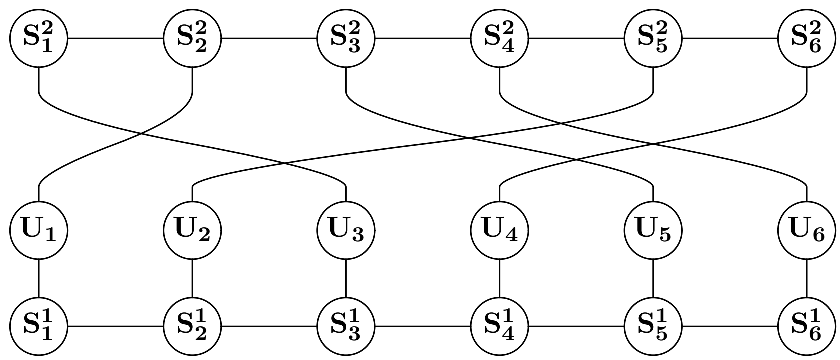


Figure 2: The cyclic graph structure underlying the turbo code of Figure 1

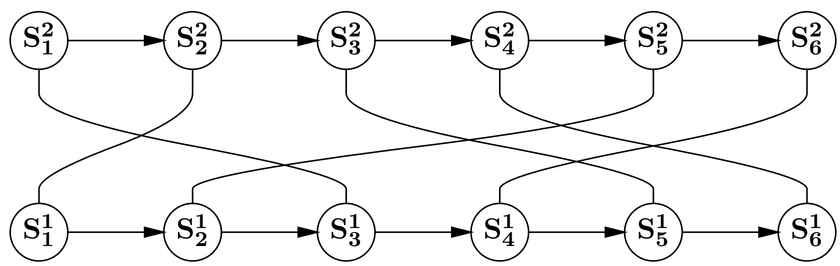


Figure 3: The underlying turbo graph for Figure 2.

for turbo-codes (e.g., Figure 1). It is well-known that in practice this decoding strategy performs well. Conversely, it is also well-known that message-passing in graphs with loops can converge to incorrect posterior probabilities (e.g., [7]). Thus, we have the "mystery" of turbo-decoding: why does a provably incorrect algorithm produce an extremely useful and practical decoding algorithm? In the remainder of this paper we take a step in understanding message-passing in graphs with loops by characterizing the distribution of cycle-lengths as a function of cycle length. The motivation is as follows: if it turns out that cycle-lengths are "long enough" then there may be a well-founded basis for believing that message-passing in graphs with cycles of the appropriate length are not susceptible to the "over-counting" problem mentioned earlier (i.e., that the effect of long loops in practice may be negligible). This is somewhat speculative and we will return to this point in Section 9. An alternative motivating factor is that the characterization of cycle-length distributions in turbo-codes is of fundamental interest by itself.

2.2 Turbo Graphs

In Figure 1 the underlying cycle structure is not affected by the **X** and **Y** nodes, i.e., they do not play any role in the counting of cycles in the graph. For simplicity they can be removed from consideration, resulting in the simpler graph structure of Figure 2. Furthermore, we will drop the directionality of the edges in Figure 2 and in the rest of the paper, since the definition of a cycle in

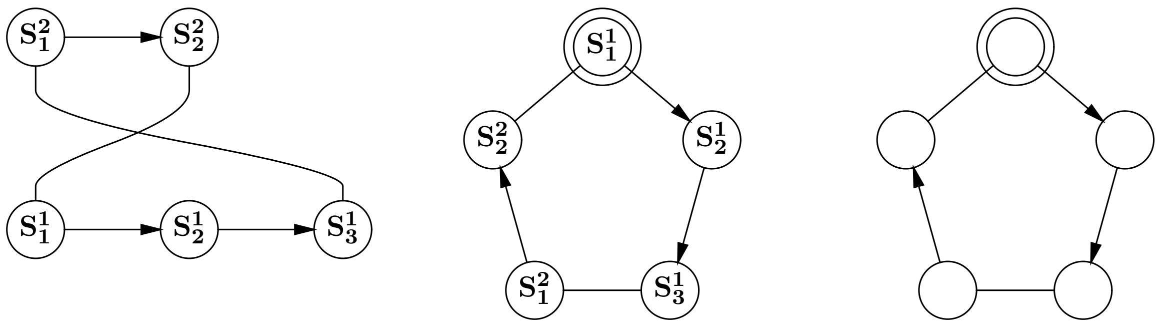


Figure 4: A simple cycle in Figure 3 and the corresponding picture (a) The simple cycle, (b) The same simple cycle, untangled, and (c) The corresponding picture.

an ADG is not a function of the directionality of the edges on the cycle.

We will focus on counting cycles of different lengths in this undirected graph. To simplify our analysis further, we initially ignore the nodes $\mathbf{U_1}$, $\mathbf{U_2}$, ..., to arrive at a *turbo graph* in Figure 3 (we will later reintroduce the $\mathbf{U}$ nodes). Formally, a turbo graph is defined as follows:

1. There are two "parallel" chains, each having n nodes. (For real turbo codes, n can be very large, e.g. $n = 64,000$.)

2. Each node is connected to one (and only one) node on the other chain and these one-to-one connections are chosen randomly, e.g., by a random permutation of the sequence $\{1, 2, \ldots, n\}$. In Section 7 we will look at another kind of permutation, the "S-random permutation".

3. A turbo graph as defined above is an *undirected* graph. But to differentiate between edges on the chains and edges connecting nodes on different chains, we label the former as being *directed* (from left to right), and the latter *undirected*. (Note: this has nothing to do with directed edges in the original ADG model, it is just a notational convenience.) So an internal node has exactly three edges connected to it: one directed edge going out of it, one directed edge going into it, and one undirected edge connecting it to a node on the other chain. A boundary node also has one undirected edge, but only one directed edge.

Given a turbo graph, and a randomly chosen node in the graph, we are interested in:

1. counting the number of simple cycles of length k which pass through this node (where a simple cycle is defined as a cycle without repeated nodes), and

2. finding the probability that this node is not on a simple cycle of length k or less, for $k = 4, 5, \ldots$ (clearly the shortest possible cycle in a turbo graph is 4).

2.3 Embedding "Pictures"

To assist our counting of cycles, we introduce the notion of a "picture." First let us look at Figure 4(a), which is a single simple cycle taken from Figure 3. When we untangle Figure 4(a), we get Figure 4(b). If we omit the node labels, we have Figure 4(c) which we call a *picture*.

Formally, a picture is defined as follows:

1. It is a simple cycle with a single distinguished vertex (the circled one in the figure).
2. It consists of both directed edges and undirected edges.
3. The number of undirected edges m is even and $m > 0$.
4. No two undirected edges are adjacent.
5. Adjacent directed edges have the same direction.

We will use pictures as a convenient notation for counting simple cycles in turbo graphs. For example, using Figure 4(c) as a template, we start from node $\mathbf{S_1^1}$ in Figure 3. The first edge in the picture is a directed forward edge, so we go from $\mathbf{S_1^1}$ along the forward edge which leads us to $\mathbf{S_2^1}$. The second edge in the picture is also a directed forward edge, which leads us from $\mathbf{S_2^1}$ to $\mathbf{S_3^1}$. The next edge is an undirected edge, so we go from $\mathbf{S_3^1}$ to $\mathbf{S_1^2}$ on the other chain. In the same way, we go from $\mathbf{S_1^2}$ to $\mathbf{S_2^2}$, then to $\mathbf{S_1^1}$, which is our starting point. As the path we just traversed starts from $\mathbf{S_1^1}$ and ends at $\mathbf{S_1^1}$, and there are no repeated nodes in the path, we conclude that we have found a simple cycle (of length 5) which is exactly what we have in Figure 4(a).

We can easily enumerate all the different pictures of length $4, 5, ..., 2n$, and use them as templates to find all the simple cycles at a node in a turbo graph. This approach is complete because any simple cycle in a graph has a corresponding picture. (To be exact, it has two pictures because we can traverse it in both directions.)

The process of finding a simple cycle using a picture as a template can also be thought of as *embedding* a picture at a node in a turbo graph. This embedding may succeed, as in our example above, or it may fail, e.g., we come to a previously-visited node other than the starting node, or we are told to go forward at the end of a chain, etc. Later in the paper we will compute the probability of successfully embedding a picture at a node in a turbo graph.

So, using the notion of pictures, our problem of counting the number of simple cycles of length k can be formulated this way:

- How many different pictures of length k are there?
- For each distinct picture, what is the probability of embedding it in a turbo graph at a randomly chosen node?

3 Counting Pictures

We wish to determine the number of different pictures of length k with m undirected edges. First, let us define two functions:

$C(a, b)$ = the number of ways of picking b disjoint edges (i.e. no two edges are adjacent to each other) from a cycle of length a, with a distinguished vertex and a distinguished clockwise direction.

$P(a, b)$ = the number of ways of picking b independent edges from a path of length a, with a distinguished endpoint.

These two functions can be evaluated by the following recursive equations:

$$C(a, b) = P(a-1, b) + P(a-3, b-1) \tag{2}$$
$$P(a, b) = P(a-1, b) + P(a-2, b-1) \tag{3}$$

and the solutions are

$$P(a, b) = \binom{a-b+1}{b} \tag{4}$$
$$C(a, b) = \binom{a-b-1}{b-1} + \binom{a-b}{b} \tag{5}$$

Thus, the number of different pictures of length k and with m undirected edges, $0 < m \le \frac{n}{2}$ (and m is even), is given by

$$\begin{aligned} N(k, m) &= 2^m \left(\binom{k-m-1}{m-1} + \binom{k-m}{m} \right) / 2 \\ &= 2^{m-1} \frac{k}{k-m} \binom{k-m}{m} \end{aligned} \tag{6}$$

where 2^m is the number of different ways to give directions to the directed edges. There is the division by two because we do not care the direction of the picture. Because of the m undirected edges, there are m segments of directed edges, with one or more edges in a segment; the edges within a segment must have a common direction (property 4 of a picture).

4 The Probability of Embedding a Picture

In this section, we derive the probability $P_n(k, m)$ of embedding a picture of length k and with m undirected edges at a node in a turbo graph with chain length n.

4.1 When $k = 2m$

Let us first consider a simple picture where the directed edges and undirected edges alternate (so $k = 2m$) and all the directed edges point in the same (forward) direction.

Let us label the nodes of the picture as X_1,X_2,Y_1,Y_2, X_3,X_4,Y_3,Y_4,. . ., X_{m-1},X_m,Y_{m-1},Y_m. We want to see if this picture can be successfully embedded, i.e. if the above nodes are a simple cycle. Let us call the chain on which X_1 resides *side 1*, and the opposite chain *side 2*. The probability of successfully embedding the picture at X_1 is the product of the probabilities of successfully following each edge of the picture, namely,

- $X_1 \to X_2$. This will fail if X_1 is the right-most node on side 1. So $p = 1 - \frac{1}{n}$.
- $X_2 \to Y_1$. Here $p = 1$.
- $Y_1 \to Y_2$. This will fail if Y_1 is the right-most node on side 2. So $p = 1 - \frac{1}{n}$.
- $Y_2 \to X_3$. X_3 is the "cross-chain" neighbor of Y_2. As there is already a connection between X_2 and Y_1, X_3 cannot possibly coincide with X_2; but it may coincide with X_1 and make the embedding fail. This gives us $p = 1 - \frac{1}{n-1}$.

  More generally, if there are $2s$ visited nodes on side 1, then s of them already have their connections to side 2. So from a node on side 2, there are only $n - s$ nodes on side 1 to go to, s of which are visited nodes. So $p = 1 - \frac{s}{n-s}$.
- $X_3 \to X_4$. Here we have two previously visited nodes (X_1,X_2). When there are $2s$ previously-visited nodes, the unvisited nodes are partitioned into up to s segments, and after we come from side 2 to side 1, if we fall on the right-most node of one of the segments, the embedding will fail: either we go off the chain, or we go to a previously-visited node. In this way, we have $1 - \frac{s+1}{n-2s} \le p \le 1 - \frac{1}{n-2s}$.
- $\cdots$
- $Y_{m-1} \to Y_m$.
- $Y_m \to X_1$. $p = \frac{1}{n-\frac{m}{2}}$. This final step ($Y_m \to X_1$) completes the cycle.

Multiplying these terms, we arrive at

$$\begin{aligned} & \frac{1}{n - \frac{m}{2}} \prod_{s=0}^{s=\frac{m}{2}} \left[\left(1 - \frac{s}{n-s}\right) \left(1 - \frac{s+1}{n-2s}\right) \right]^2 \\ \le \quad & P_n(k, m) \\ \le \quad & \frac{1}{n - \frac{m}{2}} \prod_{s=0}^{s=\frac{m}{2}} \left[\left(1 - \frac{s}{n-s}\right) \left(1 - \frac{1}{n-2s}\right) \right]^2 \end{aligned} \tag{7}$$

For large n and small m, the ratio between the upper bound and the lower bound is close to 1. For example, when $n = 64,000$, $m = 30$, the ratio is 1.0038.

4.2 The general case

The above analysis can be extended easily to the general case where:

- The direction of the directed edges in the picture can be arbitrary.
- $k \ge 2m$. (Because the m undirected edges cannot be adjacent to each other, the total number of edges k must be $\ge 2m$.)

When $k = 2m$, no two directed edges are adjacent. Equivalently, there are m segments of directed edges, and in each segment, there is only one edge. When $k > 2m$, we still have m segments of directed edges, but there is more than one edge in a segment. Suppose for $1 \le i \le m$, the ith segment of side 1 has a_i edges, and the ith segment of side 2 has b_i edges. $P_n(k,m)$ is given by:

$$
\begin{aligned}
& \frac{1}{n-\frac{m}{2}} \prod_{s=0}^{s=\frac{m}{2}} \left[\left(1-\frac{\sum_{i=1}^{s} a_i}{n-s}\right)\left(1-\frac{s+1}{n-\sum_{i=1}^{s}(a_i+1)}\right)\left(1-\frac{\sum_{i=1}^{s} b_i}{n-s}\right)\left(1-\frac{s+1}{n-\sum_{i=1}^{s}(b_i+1)}\right)\right] \\
\le\; & P_n(k,m) \qquad (8) \\
\le\; & \frac{1}{n-\frac{m}{2}} \prod_{s=0}^{s=\frac{m}{2}} \left[\left(1-\frac{\sum_{i=1}^{s} a_i}{n-s}\right)\left(1-\frac{1}{n-\sum_{i=1}^{s}(a_i+1)}\right)\left(1-\frac{\sum_{i=1}^{s} b_i}{n-s}\right)\left(1-\frac{1}{n-\sum_{i=1}^{s}(b_i+1)}\right)\right]
\end{aligned}
$$

From

$$\sum_{i=1}^{m} a_i + \sum_{i=1}^{m} b_i = k - m,$$

$$1 \le a_i,\ b_i \le 1 + (k-2m),$$

$$\sum_{i=1}^{s} a_i + \sum_{i=1}^{s} b_i \le 2s + (k-2m),$$

and

$$s \le \sum_{i=1}^{s} a_i,\ \sum_{i=1}^{s} b_i \le s + (k-2m)$$

we can simplify the bounds in Equation 8 to

$$
\begin{aligned}
& \frac{1}{n-\frac{m}{2}} \prod_{s=0}^{s=\frac{m}{2}} \left[\left(1-\frac{s+k-2m}{n-s}\right)\left(1-\frac{s+1}{n-(2s+k-2m)}\right)\right]^2 \\
\le\; & P_n(k,m) \qquad (9) \\
\le\; & \frac{1}{n-\frac{m}{2}} \prod_{s=0}^{s=\frac{m}{2}} \left[\left(1-\frac{s}{n-s}\right)\left(1-\frac{1}{n-2s)}\right)\right]^2
\end{aligned}
$$

The ratio between the upper bound and the lower bound is still close to 1. For example, when $n = 64,000, k = 100, m = 30$, the ratio is 1.0240. Given that the bounds are so close in the range of $n, k,$ and m of interest, in the remainder of the paper we will approximate $P_n(k,m)$ by the arithmetic average of the upper and lower bond.

5 The Probability of No Cycles of Length k or Less

In Section 3 we derived $N(k,m)$, the number of different pictures of length k and with m undirected cycles (Equation 6). In Section 4 we estimated $P_n(k,m)$, the probability of embedding a picture (with length k, and m undirected edges) at a node in a turbo graph with chain length n (Equation 9). With these two results, we can now determine the probability of no cycle of length k or less at a randomly chosen node in a turbo-graph of length n.

In passing, we note that the expected number of simple cycles of length k at a node in a turbo graph with chain length n is

$$\sum_{m>0, m\ even}^{m\leq\frac{k}{2}} \frac{2n}{k} \cdot N(k,m) \cdot P_n(k,m) \tag{10}$$

Let $P(\overline{\mathcal{L}_k})$ be the probability that there are no cycles of length k at a randomly chosen node in a turbo graph. Thus,

$$\begin{aligned} P(no\ cycle\ of\ length\ \leq k) &= P(\overline{\mathcal{L}}_k, \overline{\mathcal{L}}_{k-1}, \ldots, \overline{\mathcal{L}}_4) \\ &= \prod_{i=4}^{k} P(\overline{\mathcal{L}}_i \mid \overline{\mathcal{L}}_{i-1}, \ldots, \overline{\mathcal{L}}_4) \\ &\approx \prod_{i=4}^{k} P(\overline{\mathcal{L}}_i) \end{aligned} \tag{11}$$

In this independence approximation we are assuming the event that there are no cycles of length k is independent of the event that there are no cycles of length $k-1$ or lower. This is not strictly true since (for example) the non-existence of a cycle of length $k-1$ (e.g., for $k-1=4$) can make certain cycles of length k impossible (e.g., consider $k=5$). However, these cases appear to be relatively rare, leading us to believe that the independence assumption is reasonably good to first-order.

Now we estimate $P(\overline{\mathcal{L}}_k)$, the probability of no cycle of length k at a randomly chosen node. Denote the individual pictures of length k as pic_1,pic_2,..., and let $\overline{pic_i}$ mean that the ith picture fails to be embedded.

$$\begin{aligned} P(\overline{\mathcal{L}}_k) &= P(\overline{pic}_1, \overline{pic}_2, \ldots) \\ &= \prod_i P(\overline{pic}_i \mid \overline{pic}_{i-1}, \ldots, \overline{pic}_1) \\ &\approx \prod_i P(\overline{pic}_i) \\ &= \prod_{m>0, m\ even}^{m\leq\frac{k}{2}} (1-P_n(k,m))^{N(k,m)} \end{aligned} \tag{12}$$

Here we make a second independence assumption which is not strictly true. The non-existence of embedding of certain pictures (the event being conditioned on) will influence the probability of existence of embedding of other pictures. However, we are again assuming that this dependence is rather weak and that the independence assumption is again a reasonably good first-order approximation.

6 Numerical and Simulation Results

6.1 Cycle Length Distributions in Turbo Graphs

We ran a series of simulations where 200 turbo graphs of chain length $n = 64000$ are randomly generated. For each graph, we counted the simple cycles of length $k = 4, 5, \ldots, 20$, at 100 randomly chosen nodes. In total, the cycle counts at 20000 nodes are collected to to generate an empirical estimate of the true $P(no\ cycle\ of\ length\ \leq\ k)$. The theoretical estimates are derived by using the independence assumptions of Equations 11 and 12, and $P_n(k, m)$ is calculated as the arithmetic average of the the two bounds in Equation 9 (recall that the lower and upper bounds are very close). The simulation results, together with the theoretical estimates are shown in Figure 5. The difference in error is never greater than about 0.005 in probability. Note that neither the sample-based estimates nor the theoretical estimates are exact. Thus, differences between the two may be due to either sampling variation or error introduced by the independence assumptions in the estimation. The fact that the difference in errors is non-systematic (i.e., contains both positive and negative errors) suggests that both methods of estimation are fairly accurate. For comparison, in the last column of the table we provide the estimated standard deviation $\hat{\sigma}_P = \sqrt{\hat{p}(1-\hat{p})/N}$, where $\hat{p}$ is the simulation estimate. We can see that the differences between $P_{simulation}$ and $P_{theoretical}$ are within $\pm\hat{\sigma}_P$ of $P_{theoretical}$ except for the last three rows where $P_{theoretical}$ is quite small.

In Figure 6 we show a plot of the estimated probability that there are no cycles of length k or less at a randomly chosen node. There appears to be a "soft threshold effect" in the sense that beyond a certain value of k, it rapidly becomes much more likely that there are cycles of length k or less at a randomly chosen node. The location of this threshold increases as n increases (i.e., as the length of the chain gets longer).

6.2 Large-Sample Closed-Form Approximations

When n is sufficiently large, (i.e. $n \gg k$), the probability of embedding a picture (Equation 9) can simply written as

$$P_n \approx \frac{1}{n} \tag{13}$$

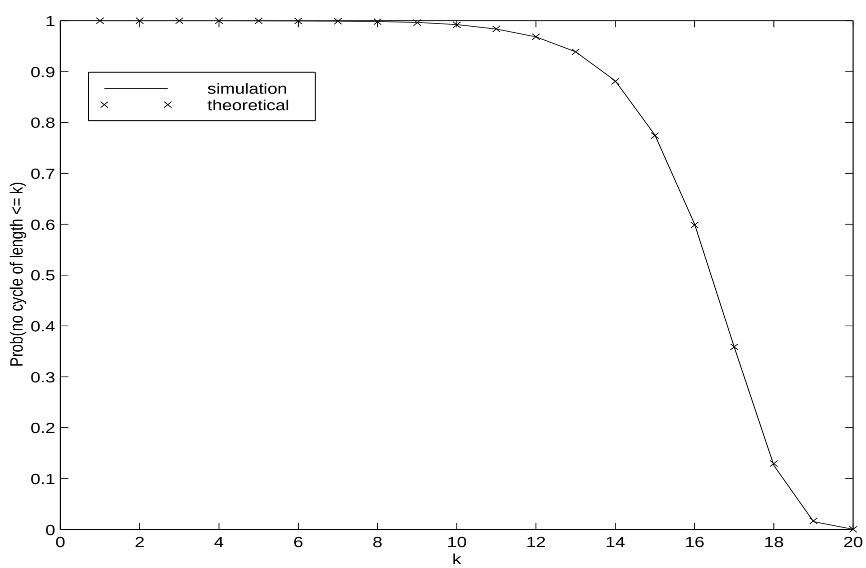
In this case, we do not differentiate between pictures with different number of undirected edges, that is, we count them together.

The total number of pictures of length k is

$$N_k \quad = \quad \sum_{m>0, m\ even}^{m \leq \frac{k}{2}} N(k, m)$$

k	$P_{simulation}$	$P_{theoretical}$	Difference	$\hat{\sigma}_P$
4	0.999950	0.999938	0.000012	0.000056
5	0.999750	0.999781	-0.000031	0.000105
6	0.999450	0.999500	-0.000050	0.000158
7	0.999100	0.999063	0.000037	0.000216
8	0.998350	0.998189	0.000161	0.000301
9	0.996650	0.996227	0.000423	0.000434
10	0.992400	0.992034	0.000366	0.000629
11	0.983750	0.983886	-0.000136	0.000890
12	0.968400	0.968456	-0.000056	0.001236
13	0.938850	0.938643	0.000207	0.001697
14	0.881800	0.880781	0.001019	0.002291
15	0.775350	0.774188	0.001162	0.002957
16	0.600550	0.598375	0.002175	0.003466
17	0.358850	0.358868	-0.000018	0.003392
18	0.125850	0.129488	-0.003638	0.002374
19	0.015500	0.016782	-0.001282	0.000908
20	0.000150	0.000279	-0.000129	0.000118

(a)



(b)

Figure 5: Theoretical vs. simulation estimates of the probability of no cycles of length k or less, as a function of k. Turbo graph chain length $n = 64,000$.

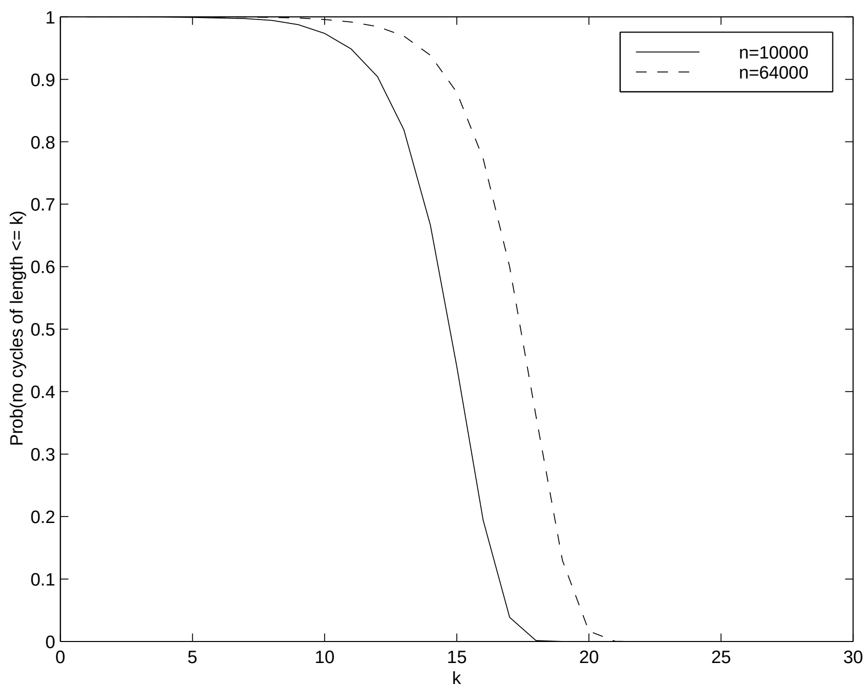


Figure 6: Approximate probability of no cycles of length k or less, as a function of k.

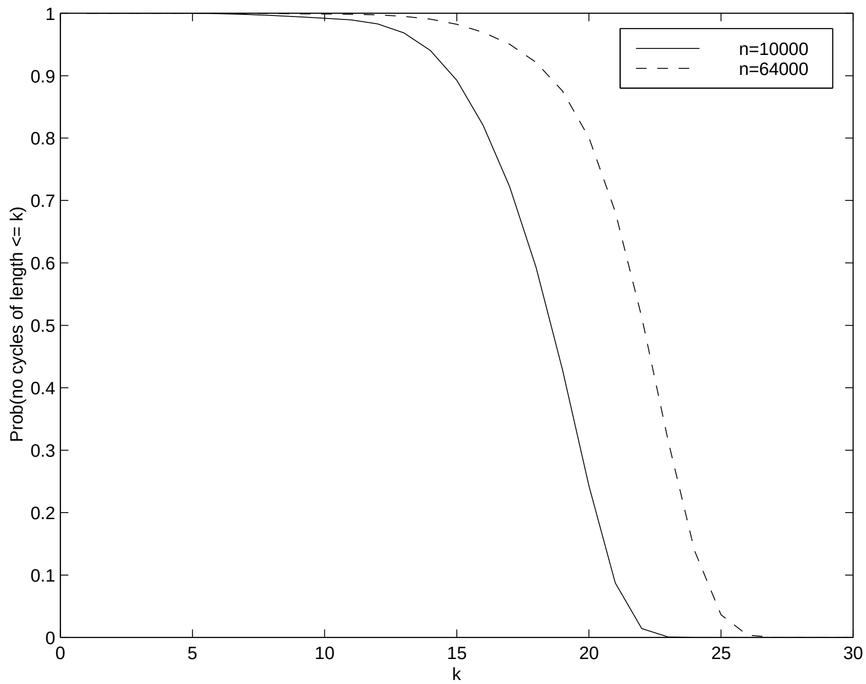


Figure 7: Probability of no cycles of length k or less, including the **U** nodes (Figures 2) in the ADG for turbo decoding, as a function of k.

$$
\begin{aligned}
&= \sum_{m>0, m\ even}^{m\leq\frac{k}{2}} 2^{m-1}\frac{k}{k-m}\left(\begin{array}{c} k-m \\ m \end{array}\right) \\
&\approx 2^{k-2} \qquad (14)
\end{aligned}
$$

The log probability of no cycle of length k is

$$
\log P(\overline{\mathcal{L}}_k) \approx \log(1-P_n)^{N_k} \approx 2^{k-2}\log(1-\frac{1}{n}) \approx -\frac{1}{n}2^{k-2} \qquad (15)
$$

From this we have

$$
\begin{aligned}
\log Prob(no\ cycle\ of\ length\ \leq k) &\approx \log\left(\prod_{i=4}^{k} P(\overline{\mathcal{L}}_i)\right) \\
&\approx \sum_{i=4}^{k}\left(-\frac{1}{n}2^{i-2}\right) \\
&= -\frac{2^{k-1}-4}{n} \qquad (16)
\end{aligned}
$$

Thus, the probability of no cycle of length k or less is approximately $e^{-\frac{2^{k-1}-4}{n}}$. This probability falls below 0.5 at $k_{0.5} = \log_2(n\log 2 + 4) + 1$. This shows us how the curve will shift to the right as n increases. Roughly speaking, to double $k_{0.5}$, we would have to double the block-length of the code from n to n^2.

6.3 Including the U Nodes

Up to this point we have been counting cycles in the turbo graph (Figure 3) where we ignore the nodes $\mathbf{U_1}$, $\mathbf{U_2}$, ... Our results can easily be extended to include these $\mathbf{U}$ nodes by counting each undirected edge (that connects nodes from different chains) as two edges.

Let $m' = \frac{m}{2},\ k' = k - \frac{m}{2}$ be the number of undirected edges and cycle length, respectively, when we ignore the $\mathbf{U}$ nodes. From $m' > 0,\ m'\ is\ even,\ m' \leq \frac{k'}{2}$, we have $m > 0,\ m\ divisible\ by\ 4,\ m \leq \frac{2k}{3}$.

Substituting these into Equation 12, we have

$$
\begin{aligned}
&\prod_{m'>0, m'\ even}^{m'\leq\frac{k'}{2}} (1-P_n(k',m'))^{N(k',m')} \\
&= \prod_{m>0, m\ divisible\ by\ 4}^{m\leq\frac{2k}{3}} \left(1-P_n(k-\frac{m}{2},\frac{m}{2})\right)^{N(k-\frac{m}{2},\frac{m}{2})} \qquad (17)
\end{aligned}
$$

Using Equation 17, we plot in Figure 7 the estimated probability of no cycles of length k or less in the graph for turbo decoding which includes the $\mathbf{U}$ nodes (Figure 2). Not surprisingly, the effect

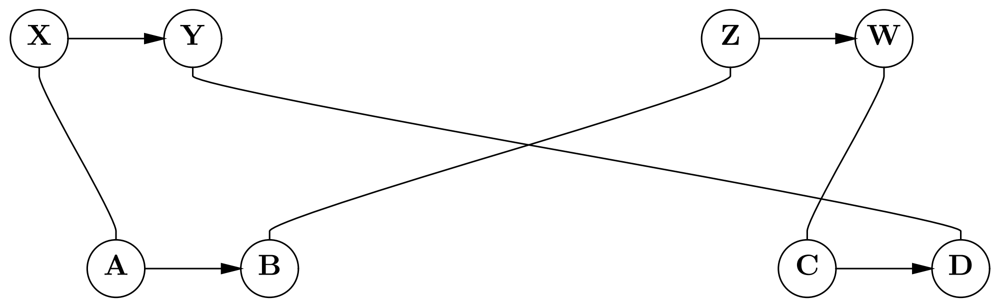


Figure 8: A cycle of length 8

is to “shift” the graph to the right, i.e., adding **U** nodes has the effect of lengthening the typical cycle.

For the purposes of investigating the message-passing algorithm, the relevant nodes on a cycle may well be those which are directly connected to a **Y** node, for example, the **U** nodes in a systematic code and any **S** nodes which are producing a transmitted codeword. The rationale for including these particular nodes (and not including nodes which are not connected to a **Y** node) is that these are the only nodes that in effect can transmit messages that potentially lead to multiple-counting. It may well be that it is the number of *these nodes* on a cycle which is relevant to message-passing algorithms rather than simply any node. Thus, for a particular code structure, the relevant nodes to count in a cycle could be redefined to be only those which have an associated **Y**. The general framework we have presented here can easily be modified to allow for such counting.

Finally, we note that various extensions of turbo-codes are also amenable to this form of analysis. For example, for the case of a turbo-code with more than two constituent encoders, one can generalize the notion of a picture and count accordingly.

7 “S-random” permutation

In our construction of the turbo graph (Figure 3) we use a random permutation, i.e. the one-to-one connections of nodes from the two chains are chosen randomly by a random permutation. In this section we look at the “S-random permutation” [8]. This is a semirandom permutation that achieves some nonrandom design objective but still retains some degree of randomness to prevent too much regularity.

Formally, the S-random permutation is a random permutation function $f(\cdot)$ on the sequence $1, 2, \ldots, n$ such that

$$\forall i, j : \mid i - j \mid \leq S \implies \mid f(i) - f(j) \mid \geq S \tag{18}$$

The S-random permutation stipulates that if two nodes on a chain are within a distance S of each other, their counterparts on the other chain cannot be within a distance S of each other. This restriction will eliminate some of the cycles occurring in a turbo graph with a purely random permutation. For example, when $S = 10$, there will be no cycles of length k =4, 5, 6 or 7.

Prob(no cycle of length k or less) for turbo graph ($n = 64000$)					
	Random	S-random permutation			
k	permutation	$S = 10$	$S = 20$	$S = 50$	$S = 100$
4	0.9999	1.0000	1.0000	1.0000	1.0000
5	0.9998	1.0000	1.0000	1.0000	1.0000
6	0.9995	1.0000	1.0000	1.0000	1.0000
7	0.9991	1.0000	1.0000	1.0000	1.0000
8	0.9982	0.9996	0.9998	0.9998	0.9998
9	0.9962	0.9983	0.9987	0.9987	0.9984
10	0.9920	0.9949	0.9945	0.9956	0.9950
11	0.9839	0.9890	0.9891	0.9877	0.9887
12	0.9685	0.9739	0.9765	0.9736	0.9748
13	0.9386	0.9460	0.9503	0.9449	0.9478
14	0.8808	0.8877	0.8920	0.8904	0.8913
15	0.7742	0.7804	0.7847	0.7858	0.7833
16	0.5984	0.6114	0.6014	0.6121	0.6006
17	0.3589	0.3671	0.3629	0.3731	0.3647
18	0.1295	0.1315	0.1289	0.1360	0.1330
19	0.0168	0.0146	0.0164	0.0184	0.0183
20	0.0003	0.0004	0.0003	0.0004	0.0008

Table 1: Estimates of the probability of no cycle of length k or less, comparing the standard random construction with the S-random construction.

In Figure 8, we show a cycle of length $k = 8$. As long as the distances of $\mid \mathbf{YZ} \mid$ and $\mid \mathbf{BC} \mid$ are large enough ($> S$), this cycle of length $k = 8$ is possible for any S. Thus, the S-random construction will produce zero probability of cycles of length k for $k < 8$. However, as we saw in Section 6, these short cycles are relatively rare for realistic turbo-codes.

For cycles of length $k \geq 8$ it is not clear what (if any) restrictions the S-random construction places on such cycles. We simulated S-random graphs and counted cycles in the same manner as described in Section 6, except that the random permutation was now carried out in the S-random fashion as described in [8]. The results in Table 1 show that changing the value of S does not appear to change the nature of the cycle-distribution. The S-random distributions of course have zero probability for $k < 8$, but for $k \geq 8$ the results from both types of permutation appear qualitatively similar, with a relatively small systematic increase in the probability of a node not having a cycle of length k for the S-random case (relative to the purely random case).

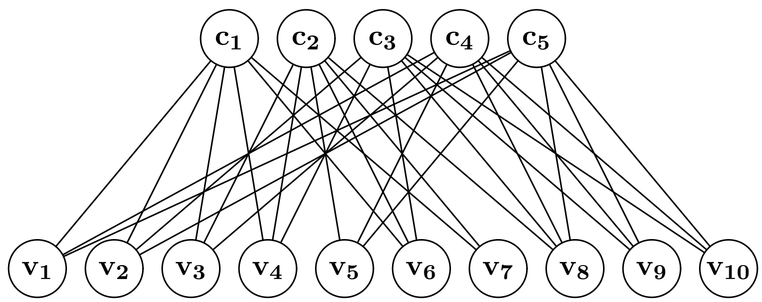


Figure 9: Graph structure of Low-Density Parity Check Codes: $d_v = 3, d_c = 6, n = 10$.

8 Low-Density Parity Check Codes

Low-Density Parity Check Codes (LDPC) [9, 10, 11] are another class of codes exhibiting characteristics and performance similar to turbo codes. The graph structure of LDPC codes is given in 9. We call it a *LDPC graph*. In this bipartite graph, at the bottom are n variable nodes $\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_n}$, and at the top are the w check nodes $\mathbf{c_1}, \mathbf{c_2}, \ldots, \mathbf{c_w}$. Each variable node has a degree of d_v, and each check node has a degree of d_c. (Obviously $nd_v = wd_c$.)

Using our notion of a *picture*, we can also analyze the distribution of cycle lengths in LDPC graphs as we have done in turbo graphs. Obviously, the cycle length must be even.

First, we define picture for an LDPC graph. Recall that in a turbo graph, the edges in a picture are labeled by *undirected, forward, backward*. For an LDPC graph, we label an edge in a picture by a number i between 1 and d_v (or between 1 and d_c) to denote that this edge is the i-th edge coming from a node.

Second, we calculate the probability of successfully embedding a picture of length $k = 2m$ at a randomly chosen node in an LDPC graph.

$$
\begin{aligned}
& P_{embed}(k = 2m) \\
= \; & 1 \cdot (1 - \frac{1}{d_c}) \cdot (1 - \frac{1}{d_v}) \\
& \cdot \left[(1 - \frac{1}{d_c})(1 - \frac{1}{n-1})\right] \\
& \cdot \left[(1 - \frac{1}{d_v})(1 - \frac{1}{c-1})\right] \\
& \cdot \left[(1 - \frac{1}{d_c})(1 - \frac{2}{n-1})\right] \\
& \cdot \left[(1 - \frac{1}{d_v})(1 - \frac{2}{c-1})\right] \\
& \cdots \\
& \cdot \left[(1 - \frac{1}{d_c})(1 - \frac{m-2}{n-1})\right]
\end{aligned} \tag{19}
$$

$$
\begin{aligned}
&\cdot\left[(1-\frac{1}{d_v})(1-\frac{m-2}{c-1})\right] \\
&\cdot\left[(1-\frac{1}{d_c})\frac{1}{n-1}\right] \\
= \quad & \frac{1}{n-1}\left(1-\frac{1}{d_c}\right)^m\left(1-\frac{1}{d_v}\right)^{m-1}\prod_{i=0}^{m-2}\left[\left(1-\frac{i}{n-1}\right)\left(1-\frac{i}{c-1}\right)\right]
\end{aligned}
$$

Then, we calculate the number of different pictures of length $k = 2m$.

$$
N(k = 2m) = \frac{1}{2}d_c^m d_v^m \tag{20}
$$

And finally we calculate the probability of no cycle of length $k = 2m$ at a randomly chosen node in a LDPC graph.

$$
\begin{aligned}
& Prob(no\ cycle\ of\ length\ k = 2m\ or\ less) \\
\approx \quad & \prod_{i=4,i\ even}^{k} Prob(no\ cycle\ of\ length\ i) \\
\approx \quad & \prod_{i=4,i\ even}^{k} (1 - P_{embed}(i))^{N(i)}
\end{aligned} \tag{21}
$$

We also ran a number of simulations in which we randomly generate some LDPC graphs and count the cycles in them. We plot in Figures 11 and 10 the results of the simulation and the theoretical estimates from Equation 21 for $n = 15000$ and 63000.

From the simulation results we see that the LDPC curve is qualititatively similar in "shape" to the turbo-graph curves earlier but has been shifted to the left, i.e., there is a higher probability of short cycles in an LDPC graph than in a turbo graph, for the typical parameters we have chosen to work with. This is not surprising, since the branching factor in a turbo-graph is 3 (each node is connected to 3 neighbors) while the average branching factor in an LDPC graph (as analyzed with $d_c = 5, d_v = 3$) is 4.

The independence assumptions clearly are not as accurate in the LDPC case as they were for the turbo graphs. Recall that we make two separate independence assumptions in our analysis, namely that

1. the event that there is no cycle of length k, is independent of the event that there are no cycles of length $k-1$ or lower, and
2. the event that a particular picture cannot be embedded at a randomly chosen node is independent of the event that other pictures cannot be embedded.

We can check the accuracy of the first independence assumption readily by simulation. We ran a number of simulations to count cycles in randomly generated graphs. From the simulation data, we estimate the marginal probability $P(\overline{\mathcal{L}}_k)$, and the joint probability $P(\overline{\mathcal{L}}_k)P(\overline{\mathcal{L}}_{k+1})$. To see if our independence approximation is valid, we compare the product of estimated marginal probabilities with the estimated joint probability.

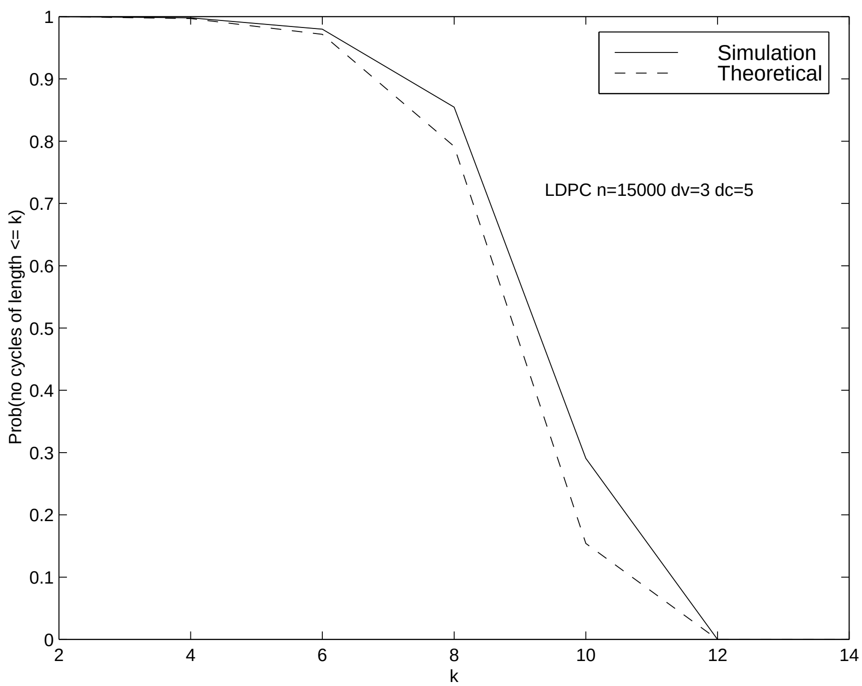


Figure 10: Approximate probability of no cycles of length k or less in an LDPC graph with $n = 15000$, as a function of k.

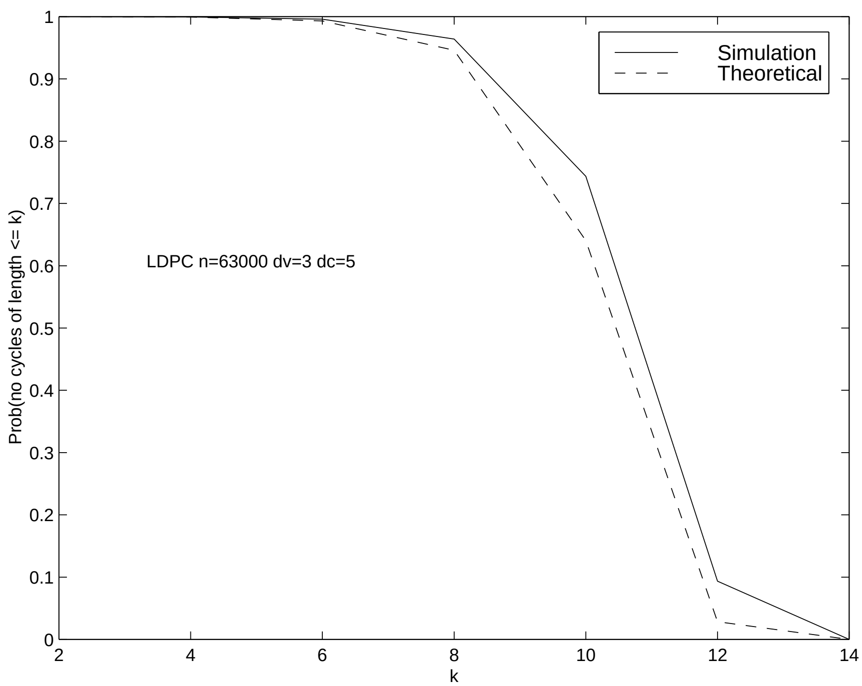


Figure 11: Approximate probability of no cycles of length k or less in an LDPC graph with $n = 63000$, as a function of k.

k	$P(\overline{\mathcal{L}}_k)P(\overline{\mathcal{L}}_{k+1})$	$P(\overline{\mathcal{L}}_k, \overline{\mathcal{L}}_{k+1})$	Difference
1	1.000000	1.000000	0.000000
2	1.000000	1.000000	0.000000
3	0.999950	0.999950	0.000000
4	0.999750	0.999750	0.000000
5	0.999500	0.999500	0.000000
6	0.999350	0.999350	0.000000
7	0.998900	0.998900	0.000000
8	0.997551	0.997550	0.000001
9	0.994007	0.994000	0.000007
10	0.986988	0.987050	-0.000062
11	0.975836	0.975850	-0.000014
12	0.954670	0.954500	0.000170
13	0.910886	0.910800	0.000086
14	0.826067	0.825350	0.000717
15	0.681002	0.679800	0.001202
16	0.460932	0.463650	-0.002718
17	0.213367	0.212600	0.000767
18	0.046814	0.046650	0.000164
19	0.002129	0.001350	0.000779

Table 2: The independence between $P(\overline{\mathcal{L}}_k)$ and $P(\overline{\mathcal{L}}_{k+1})$ in turbo graphs with chain length $n = 64000$.

k	$P(\overline{\mathcal{L}}_{2k})P(\overline{\mathcal{L}}_{2k+2})$	$P(\overline{\mathcal{L}}_k, \overline{\mathcal{L}}_{k+1})$	Difference
1	0.999542	0.999542	0.000000
2	0.995460	0.995458	0.000002
3	0.963715	0.963708	0.000007
4	0.746716	0.746771	-0.000055
5	0.097712	0.097333	0.000379

Table 3: The independence between $P(\overline{\mathcal{L}}_k)$ and $P(\overline{\mathcal{L}}_{k+1})$ in LDPC graphs with $n = 63000, d_v = 3, d_c = 5$.

Table 2 provides the comparison for turbo graphs for $n = 64000$. The product of marginal probabilities is close to the joint probability, so the independence approximation appears to be valid for turbo graphs. Table 3 gives a similar results for LDPC. Thus, we conclude that the first independence assumption (that the non-occurrence of cycles of length k is independent of the non-occurrence of cycles of length $k-1$ of less) appears to be quite accurate for both turbo graphs and LDPC graphs.

Since assumption 2 is the only other approximation being made in the analysis of the LDPC graphs, we can conclude that it is this approximation which is less accurate (given that the approximation and simulation do not agree so closely overall for LDPC graphs). Recall that the second approximation is of the form:

$$\begin{aligned} P(\overline{\mathcal{L}}_k) &= P(\overline{pic}_1, \overline{pic}_2, \ldots) \\ &= \prod_i P(\overline{pic}_i \mid \overline{pic}_{i-1}, \ldots, \overline{pic}_1) \\ &\approx \prod_i P(\overline{pic}_i) \end{aligned}$$

This assumption can fail for example when two pictures have the first few edges in common. If one fails to be embedded on one of these common edges, then the other will fail too. So the best we can hope from this approximation is that because there are so many pictures, these dependence effects will cancel out. In other words, we know that

$$P(\overline{pic}_i) \neq P(\overline{pic}_i \mid \overline{pic}_{i-1}, \ldots, \overline{pic}_1)$$

but we hope that

$$P(\overline{pic}_1, \overline{pic}_2, \ldots) \approx \prod_i P(\overline{pic}_i).$$

One possible reason for the difference between the LDPC case and the turbo case may be the fact that for turbo graphs, in the composition of the probability of embedding a picture in turbo graph,

$$P_n(k,m) \approx \frac{1}{n - \frac{m}{2}} \prod_{s=0}^{s=\frac{m}{2}} \left[\left(1 - \frac{s}{n-s}\right)\left(1 - \frac{1}{n-2s)}\right)\right]^2$$

the term $\frac{1}{n-\frac{m}{2}}$ is the most important, i.e., all other terms one are nearly 1. So even if two pictures share many common edges and become "dependent," as long as they do not share that most important edge, they can be regarded as effectively independent.

In contrast, for LDPC graphs, the contributions from the edges to probability tend to be more "distributed," or even. Each edge contributes a $\left(1 - \frac{1}{d_c}\right)$ term or a $\left(1 - \frac{1}{d_v}\right)$ term. No single edge dominates

$$P_{embed}(k = 2m) = \frac{1}{n-1}\left(1 - \frac{1}{d_c}\right)^m \left(1 - \frac{1}{d_v}\right)^{m-1} \prod_{i=0}^{m-2} \left[\left(1 - \frac{i}{n-1}\right)\left(1 - \frac{i}{c-1}\right)\right],$$

and, thus, the "effective independence" does not hold as in the case of turbo-graphs.

9 Discussion

We have shown that randomly chosen nodes in turbo graphs are relatively rarely on a cycle of length 10 or less, but are highly likely to be on a cycle of length 20 or greater (for a block length of 64000). We can speculate as to what this may imply about the accuracy of the iterative message-passing algorithm. It is possible to show that there is a well-defined "distance effect" in message propagation for typical ADG models, namely that the probability that a particular observed $\mathbf{Y}$ value will effect a change in the bit decision of any $\mathbf{U}$ node is inversely proportional to the distance (path length) between the $\mathbf{Y}$ node and the $\mathbf{U}$ node in the underlying graph [12]. The further away the $\mathbf{Y}$ node is from the $\mathbf{U}$ node, the less effect it will have (on average) on the posterior probability of $\mathbf{U}$. Furthermore, the less noisy the $\mathbf{Y}$ information (for decoding, the higher the signal-to-noise ratio), the less likely it is that $\mathbf{U}$ will change given a change in the value of $\mathbf{Y}$ [12]. Thus, under appropriate signal-to-noise settings and given sufficiently lengthy loops, it is plausible that the information which is in danger of being counted multiple times tends to effectively "die out" before causing the algorithm to make an error. Of course, for any node $\mathbf{U}$ there are an exponentially increasing number of nodes $\mathbf{Y}$ at a distance d away as d increases, so that this argument is by no means sufficient on its own to explain the apparent accuracy of iterative decoding algorithms. However, the results in this paper do suggest that the cycles in turbo graphs have a definite distributional character and we hope that this information can be used to further understand the workings of iterative decoding.

We also note that we have conducted experiments with the message-passing algorithm on graphs with very short loops (e.g., with just 6 nodes in total) where the exact solution can also be computed. Typically, the posterior probabilities calculated by the iterative algorithm on these small graphs can be very far away from their true exact values; however, the bit decisions (for binary nodes) seem to be relatively accurate on average, at least for many randomly chosen graphs. This might suggest that cycle-length does not play a role in iterative decoding, given that iterative decoding appears to work on some graphs with very short loops. However, we suspect that the results on small graphs may not hold much insight into what is going on with realistically-sized turbo-codes, and in particular, that the cycle-length phenomenon may well be playing an important role in the success of turbo-decoding.

Acknowledgments

The authors are grateful to R. J. McEliece and the coding group at Caltech for useful discussions and feedback.

References

1. C. Berrou, A. Glavieux, and P. Thitimajshima (1993). Near Shannon limit error-correcting coding and decoding: Turbo codes. *Proceedings of the IEEE International Conference on Communications.* pp. 1064-1070.

2. R.J. McEliece, D.J.C. MacKay and J.-F. Cheng (1998). Turbo Decoding as an Instance of Pearl's 'Belief Propagation' Algorithm. *IEEE Journal on Selected Areas in Communications,* SAC-16(2):140-152.

3. F. R. Kschischang, B. J. Frey (1998). Iterative Decoding of Compound Codes by Probability Propagation in Graphical Models. *IEEE Journal on Selected Areas in Communications,* SAC-16(2):219-230.

4. B. J. Frey (1998). *Graphical Models for Machine Learning and Digital Communication.* MIT Press: Cambridge, MA.

5. J. Pearl (1988), *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann Publishers, Inc., San Mateo, CA.

6. L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, 'Optimal decoding of linear codes for minimizing symbol error rate,' *IEEE Transactions on Information Theory*, 20:284–287, 1974.

7. R. J. McEliece, E. Rodemich, and J. F. Cheng, 'The Turbo-decision Algorithm,' in *Proc. Allerton Conf. on Comm., Control, Comp.*, 1995.

8. S. Dolinar and D. Divsalar (1995), *Weight Distributions for Turbo Codes Using Random and Nonrandom Permutations.* TDA Progress Report 42-121 (August 1995), Jet Propulsion Laboratory, Pasadena, California.

9. R. G. Gallager (1963), *Low-Density Parity-Check Codes.* Cambridge, Massachusetts: M.I.T. Press.

10. D.J.C. MacKay, R.M. Neal (1996), *Near Shannon Limit Performance of Low Density Parity Check Codes*, published in *Electronics Letters*, also available from `http://wol.ra.phy.cam.ac.uk/`.

11. T. Richardson, R. Urbanke (1998), *The Capacity of Low-Density Parity Check Codes under Message-Passing Decoding*, preprint, available at *http://cm.bell-labs.com/who/tjr/pub.html*.

12. X. Ge and P. Smyth, '*Distance effects in message propagation*', in preparation.

13. M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, D. A. Spielman (1998), 'Analysis of Low Density Codes and Improved Designs Using Irregular Graphs,' in *Proceedings of the 30th ACM STOC.* Also available online at *http://www.icsi.berkeley.edu/ luby/index.html*.