RZ 3346 (# 93392) 05/04/01
Electrical Engineering 5 pages

Research Report

Progressive Edge-Growth Tanner Graphs

Xiao-Yu Hu, Evangelos Eleftheriou, and Dieter-Michael Arnold

IBM Research
Zurich Research Laboratory
8803 Rüschlikon
Switzerland

Progressive Edge-Growth Tanner Graphs

Xiao-Yu Hu, Evangelos Eleftheriou, and Dieter-Michael Arnold

IBM Research, Zurich Research Laboratory, 8803 Rüschlikon, Switzerland

Abstract

We propose a general method for constructing Tanner graphs having a large girth by progressively establishing edges or connections between symbol and check nodes in an edge-by-edge manner, known as progressive edge-growth (PEG) construction. Simulation results show that the resulting low-density parity-check (LDPC) codes of PEG Tanner graphs significantly outperform randomly constructed ones.

Introduction: Codes on graphs [1] have attracted considerable attention owing to their capacity-approaching performance and low-complexity iterative decoding. The prime examples of codes on graphs are low-density parity-check codes (LDPC) [2]. Although random graphs have been used to construct LDPC codes with impressive performance [3], there is no guarantee that any given random graph also defines a good code having a respectable shortest cycle (girth) that facilitates iterative decoding, especially for relatively short block lengths. Here we present a simple but efficient method for constructing Tanner graphs having a large girth in a best-effort sense by progressively establishing edges between symbol and check nodes in an edge-by-edge manner. Given the number of symbol nodes n, the number of check nodes m, and the symbol-node degree sequence of the graph, we start an edge-selection procedure such that the placement of a new edge on the graph has as small an impact on the girth as possible. After a best-effort edge has been determined, we update the graph with this new edge, and proceed with the placement of a next edge.

Notations: An LDPC code is a linear code defined by a sparse parity check matrix H having dimension $m \times n$. A bipartite graph with m check nodes in one class and n symbol nodes in the other can be created using H as the integer-valued incidence matrix for the two classes. Such a graph is also called Tanner graph [4]. Formally, a Tanner graph is denoted as (V, E) with V the set of vertices (nodes), i.e. $V = V_c \cup V_s$, where $V_c = \{c_0, c_1, \ldots, c_{m-1}\}$ is the set of check nodes and $V_s = \{s_0, s_1, \ldots, s_{n-1}\}$ the set of symbol nodes. E is the set of edges such that $E = V_c \times V_s$ with edge $(c_i, s_j) \in E$ if and only if $h_{i,j} \neq 0$, $h_{i,j} \in H$, $0 \leq i \leq m-1, 0 \leq j \leq n-1$. A Tanner graph is called *regular* if every symbol node participates in d_s check nodes and every check node involves d_c symbol nodes; otherwise it is called *irregular*. Denote the symbol degree sequence by $D_s = \{d_{s_0}, d_{s_1}, \ldots, d_{s_{n-1}}\}$, in which d_{s_j} is the degree of symbol node s_j, and let edges E be partitioned in terms of V_s as $E = E_{s_0} \cup E_{s_1} \cup \cdots \cup E_{s_{n-1}}$, with E_{s_j} containing all edges incident on symbol node s_j. Furthermore, denote the k-th edge incident on s_j by $E_{s_j}^k$, $0 \leq k \leq d_{s_j} - 1$.

For a given symbol node s_j, define its *neighbor within depth* l, $\mathcal{N}_{s_j}^l$, as the set consisting of all check nodes reached by a tree spreading from symbol node s_j within depth l, as shown in the example in Fig. 1. Its complementary set, $\bar{\mathcal{N}}_{s_j}^l$, is defined as $V_c \backslash \mathcal{N}_{s_j}^l$, or equivalently $\bar{\mathcal{N}}_{s_j}^l \cup \mathcal{N}_{s_j}^l = V_c$.

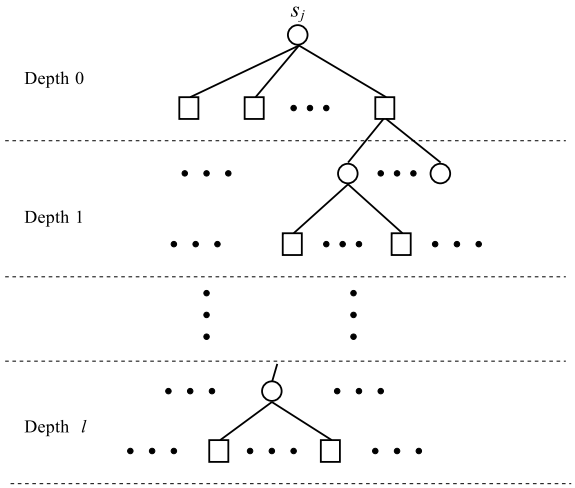

Figure 1: A graphical view of neighbor $\mathcal{N}_{s_j}^l$ within depth-l of symbol node s_j.

To efficiently compute $\mathcal{N}_{s_j}^l$ and its counterpart $\bar{\mathcal{N}}_{s_j}^l$, we set an indicator $\mathcal{I}_{c_i}$ for each check node c_i taking on values from the set $\{0,1\}$. The indicator set $\mathcal{I}$ is initialized to 0. As the tree originating in s_j proceeds to depth l, the indicators of all check nodes included in the spanning tree are set to 1, indicating that these nodes belong to $\mathcal{N}_{s_j}^l$. Likewise, $\bar{\mathcal{N}}_{s_j}^l$ is obtained by checking whether the indicator set $\mathcal{I}_{c_i}$ equals 0. The computational complexity of this procedure is approximately linear in m.

In graph theory, *girth* g refers to the length of the shortest cycle in a graph. For each symbol node s_j, we define a local girth g_{s_j} as the length of the shortest cycle passing through that symbol node. The set of local girths $\{g_{s_j}\}$ is referred to as girth histogram. It follows, by definition, that $g = \min_j\{g_{s_j}\}$.

The PEG algorithm: Constructing a Tanner graph with the largest possible girth is a rather difficult combinatorial problem. Nevertheless, a sub-optimal algorithm to construct a Tanner graph with a relatively large girth is possible. One such algorithm is the progressive edge-growth algorithm, in which the local girth of a symbol node is maximized whenever a new edge is added to this symbol node. Suppose we have finished constructing edges of the first j symbol nodes on a Tanner graph, i.e., edges $E_{s_0} \cup E_{s_1} \cup \cdots \cup E_{s_{j-1}}$ have been established. Let g^t be the temporary girth under the current graph setting with edges $E_{s_0} \cup E_{s_1} \cup \cdots \cup E_{s_{j-1}}$. In other words, $g^t = \min\{g_{s_0}, g_{s_1}, \ldots, g_{s_{j-1}}\}$. The problem of constructing a graph having a large girth lies in how to select the edge set E_{s_j} of symbol node s_j such that adding these new edges to the current graph setting does not impair the current g^t too strongly. This boils down to optimizing d_{s_j} edges of E_{s_j} to maximize the local girth g_{s_j} because, if adding E_{s_j} to the current graph results in a cycle shorter than g^t, this new short cycle must pass through symbol node s_j. Unfortunately the complexity of exhaustively searching for the optimum set E_{s_j} is governed by $\binom{m}{d_{s_j}}$, where $\binom{m}{d_{s_j}}$ is a binomial coefficient. Thus, we propose a best-effort algorithm known as progressive edge-growth, in which d_{s_j} edges of E_{s_j} are added to the current graph on an edge-by-edge basis, and the length of the shortest cycle passing through symbol node s_j is maximized whenever a new edge originating in s_j is being added. This can be accomplished simply by first expanding the tree originating in symbol node s_j up to depth l each time a new edge of s_j is being determined, such that $\bar{\mathcal{N}}_{s_j}^l \neq \varnothing$ but $\bar{\mathcal{N}}_{s_j}^{l+1} = \varnothing$ or the cardinality of $\mathcal{N}_{s_j}^l$ stops increasing but is smaller than m, and then placing an edge between s_j and a check node selected from the set $\bar{\mathcal{N}}_{s_j}^l$. In this way, the shortest cycle passing through this new edge under the current graph setting is guaranteed to be no shorter than $2(l+2)$. We summarize the proposed algorithm as follows.

Progressive Edge-Growth Algorithm:
for $j = 0$ to $n-1$ **do**
begin
  for $k = 0$ to $d_{s_j} - 1$ **do**
  begin
    if $k = 0$
      $E_{s_j}^0 \longleftarrow$ edge (c_i, s_j), where $E_{s_j}^0$ is the first edge incident to s_j and c_i is one check node such that it has the lowest check degree under the current graph setting $E_{s_0} \cup E_{s_1} \cup \cdots \cup E_{s_{j-1}}$.
    else

expanding a tree from symbol node s_j up to depth l under the current graph setting such that $\bar{\mathcal{N}}_{s_j}^{l} \neq \varnothing$ but $\bar{\mathcal{N}}_{s_j}^{l+1} = \varnothing$, or the cardinality of $\mathcal{N}_{s_j}^{l}$ stops increasing but is less than m, then $E_{s_j}^{k} \longleftarrow$ edge (c_i, s_j), where $E_{s_j}^{k}$ is the k-th edge incident to s_j and c_i is one check node picked from the set $\bar{\mathcal{N}}_{s_j}^{l}$ having the lowest check-node degree.

end

end

There is a subtle point in the PEG algorithm that needs further comment. Whenever we encounter multiple possible choices for connecting to symbol node s_j, i.e. multiple check nodes exist in $\bar{\mathcal{N}}_{s_j}^{l}$, we select the one having the smallest number of incidence edges under the current graph setting. Such a selective strategy renders the resulting PEG Tanner graph as check-node degree-uniform as possible.

Even so, we may still face a situation in which multiple choices exist because multiple check nodes in $\bar{\mathcal{N}}_{s_j}^{l}$ might have the same lowest degree, particularly at the starting period of the construction. There are two main approaches to solve this problem. The first is to randomly select one of these check nodes. The other is to always select one according to its position in the order of $c_0, c_1, \ldots, c_{m-1}$. For instance, we can first sort the check nodes in $\bar{\mathcal{N}}_{s_j}^{l}$ that have the same lowest degree according to their subscripts, and then always pick the first one. Here we adopt the first approach; however, note that the second may also be of interest because of its deterministic nature.

Performance: We study the performance of PEG Tanner graphs applied to binary LDPC codes by means of Monte Carlo simulations. For comparison purposes, we use the rate-1/2 ($n = 504, m = 252$) code of MacKay in [5], which is a regular Tanner graph with $d_s = 3, d_c = 6$. Currently, this code is one of the best codes with these parameters. A PEG Tanner graph of 504 symbol and 252 check nodes is generated with uniform degree 3 for each symbol node. The resulting graph is nearly check-node uniform with degree 6, except for eight check nodes with a degree of 7, and eight with a degree of 5. We also use a randomly constructed rate-1/2 (504, 252) code, in which the degree of symbol nodes is 3 and the degree of check nodes is made as uniform as possible. In constructing such a random graph, we made sure that no 4-cycles were generated.

Figure 2 compares the girth histogram of the PEG, MacKay, and random graph codes. In the PEG Tanner graph, each symbol node has a local girth of 8, except for three symbol nodes with a local girth of 10. In the MacKay's code, 63% of the symbol nodes have a local girth of 6 and 37% one of 8. In the random graph, 79% of the symbol nodes have a local girth of 6 and 21% one of 8. The average local girth of these three graphs is 8.01, 6.74, and 6.42, respectively. Clearly, based on the girth histogram, the PEG Tanner graph has an advantage over its two counterparts.

Figure 3 compares the bit and block error rates for the three codes after 80 iterations, and reveals that the random graph is much worse than the other two codes, perhaps mainly because of its poor girth histogram. We observe that the PEG Tanner graph is always slightly better than the MacKay's code. With 80 iterations and at a block error rate of 5×10^{-5}, the PEG Tanner graph outperforms the MacKay's code by 0.2 dB. The significance of the PEG Tanner graph should not be underestimated considering that—to the best of our knowledge—MacKay's codes still are the best codes for short and medium block lengths.

Conclusions: A general method for constructing Tanner graphs having a large girth has been presented. The main principle of this construction is to optimize the placement of a new

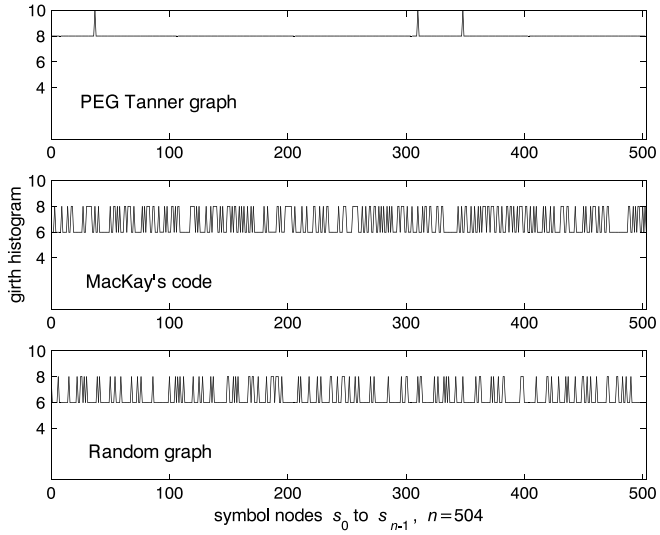

Figure 2: Girth histograms of a PEG Tanner graph, MacKay's code, and a random graph, with parameters $n = 504, m = 252, d_s = 3, d_c = 6$.

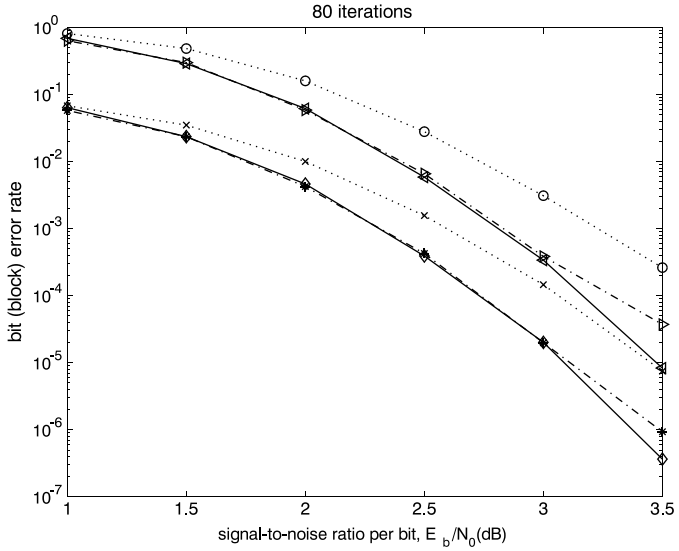

Figure 3: Bit and block error rate of the PEG, MacKay's, and the random codes, with parameters $n = 504, m = 252, d_s = 3, d_c = 6$.

—◇— PEG Tanner graph, bit error
- ·∗- · Mackay code, bit error
· · · × · · · random graph, bit error
—◁— PEG Tanner graph, block error
- ·▷- · MacKay code, block error
· · · ○ · · · random graph, block error

edge, connecting a particular symbol node to a specific check node on the graph such that the largest possible local girth is achieved. In this way, the underlying graph grows in an edge-by-edge manner, optimizing each local girth, and is referred to as a progressive edge-growth

Tanner graph. Simulation results have shown that using the PEG algorithm for constructing short block-length LDPC codes results in a significant improvement compared to randomly constructed codes.

References

[1] Special issue on codes and graphs and iterative algorithms, IEEE Trans. Inform. Theory, 2001, 47, (2), pp. 493-849

[2] GALLAGHER, R.G.: 'Low density parity check codes' (MIT Press, Cambridge, MA, 1963)

[3] MACKAY, D.J.C.: 'Good error-correcting codes based on very sparse matrices', IEEE Trans. Inform. Theory, 1999, 46, (8), pp. 399-431

[4] TANNER, R.M.: 'A recursive approach to low complexity codes', IEEE Trans. Inform. Theory, 1981, 27, (9) pp. 553-547

[5] MACKAY, D.J.C.: 'Online database of low-density parity-check codes', available at `http://wol.ra.phy.cam.uk /mackay/codes/data.html`.