

# Generating random Tanner-graphs with large girth

Mohsen Bayati\*, Raghunandan Keshavan\*, Andrea Montanari\*<sup>†</sup>, Sewoong Oh\*, and Amin Saberi<sup>‡§</sup>

\*Department of Electrical Engineering

<sup>†</sup>Department of Statistics

<sup>‡</sup>Department of Management Science and Engineering

<sup>§</sup>Institute for Computational and Mathematical Engineering

Stanford University, Stanford, CA 94305

{bayati,raghuram,montanar,swoh,saberi}@stanford.edu

**Abstract**—We present a simple and efficient algorithm for randomly generating Tanner-graphs with given symbol-node and check-node degrees and without small cycles. These graphs can be used to design high performance Low-Density Parity-Check (LDPC) codes.

Our algorithm generates a graph by sequentially adding the edges to an empty graph. Recently, these types of *sequential methods for counting and random generation* have been very successful [35], [18], [11], [7], [4], [6].

## I. INTRODUCTION

We present an efficient algorithm for generating random bipartite graphs with given node degrees and with cycles of size larger than a constant  $k$ <sup>1</sup>. The main motivation for this work comes from the design of high performance Low-Density Parity-Check (LDPC) codes [31].

Our algorithm is based on the algorithm S presented in [5]. For positive integers  $m, n, k$ , Algorithm S generates a random graph with  $n$  vertices and  $m$  edges that has no cycles with length less than or equal to  $k$  using  $O(n^2m)$  operations. Algorithm S starts with an empty graph and sequentially adds  $m$  edges between pairs of non-adjacent vertices. In every step of the procedure, an edge can be added between two distinct vertices  $i$  and  $j$  that are of distance at least  $k$ . The probability of adding an edge between  $i$  and  $j$ , denoted by  $p_{ij}$ , changes in every step of the algorithm. In order to get a uniform sampling,  $p_{ij}$  should be proportional to the number of extensions of the current graph to graphs with  $m$  edges that contain  $(ij)$  and have no small cycles. Algorithm S estimates the number of such valid extensions by computing the expected number of small cycles if the rest of edges are added uniformly at random (see Sections 3-4 of [5] for more details).

The main objective of this paper is to provide the extension of Algorithm S from [5] to graphs with given degrees.

From a theoretical perspective, our work and [5] are related to the following problem. Given a graph property  $P$  that is preserved by removal of any edge from the graph. A *random maximal  $P$ -graph* is obtained from  $n$  isolated vertices by randomly adding those edges (at each stage choosing uniformly among edges whose inclusion would not destroy property  $P$ ) until no further edges can be added. The question of finding the number of edges of a random maximal  $P$ -graph for several properties  $P$  is well studied [33], [16],

[38], [8], [27]. In particular, when  $P$  is the property that the graph has girth greater than  $k$ , [27] shows that the above process of sequentially growing the graph leads to graphs with  $m = O(n^{1+\frac{1}{k-1}} \log n)$  edges.

Unfortunately, these random maximal  $P$ -graphs may have distribution that are far from uniform. In fact it has been shown (e.g. [36]) that when  $P$  is the property of having no triangle, the maximal triangle free graphs are close to bipartite. The analysis of [5] show that the Algorithm S guarantee asymptotically uniform distribution at the expense of reducing the number of edges to  $m = O(n^{1+\frac{1}{2k(k+3)}})$ .

Recently, sequential algorithms have been shown, empirically ([11], [7]) and theoretically ([35], [18], [6], [4]), to be very successful for designing fast algorithms for counting and generating random graphs with given degrees.

### A. Application in designing LDPC Codes

It has been shown that LDPC codes can approach Shannon capacity asymptotically for large size codes, when their associated graph representations (Tanner graphs) are selected uniformly at random from the set of bipartite graphs with a properly optimized degree sequences [12], [21]. However, in practice, the maximum graph size is between  $10^3$  and  $10^5$  (depending on the delay sensitivity and on the hardware constraints). In this range, it is well known that the existence of a small number of subgraphs with a certain structure (in particular, small cycles) spoil the code performances [28], [30], [20].

Different approaches have been developed within the coding theory community to deal with this problem. For example, deterministic constructions of graph sequences with large girth [26], [32] have been studied. However, numerical studies have shown that known deterministic constructions can have poor performance [22]. From a theoretical point of view, no deterministic graph sequence is known that asymptotically outperforms random graphs.

One can also stick to random constructions and grow the graph by adding random edges sequentially while avoiding short cycles. This method has been very popular in practice and is known by the name of *progressive edge growth* (PEG) algorithm [14]. We will describe the main intuition behind PEG and show its limitations with respect to two standard performance measures for the codes: (i) bit error rate or

<sup>1</sup>Our algorithm can be applied to non-bipartite graphs as well.

expected fraction of wrong bits; and (ii) block error rate or probability that at least one bit in the message was received incorrectly.

Let  $C_{\text{iter}}$  be the maximum rate achievable by random LDPC codes (empirically  $C_{\text{iter}}$  is indistinguishable from the channel capacity). It is known that uniformly random graphs contain a random number (of order  $O(1)$ ) of cycles of size  $k$  or smaller. These cycles are responsible for non-vanishing block error probability that is bounded away from 0 at small noise. The main goal of PEG is to reduce this error, to a value that vanishes with  $k$ , by removing the cycles of length up to  $k$ . But the final distribution of PEG is not necessarily uniform which may affect the other performance measure (bit error rate). In fact preliminary simulations suggest that our new algorithm produces codes with lower bit error rate.

In this paper we define the first code generation algorithm that overcomes both problems. We show that there exists a graph sequence that (1) can be generated efficiently; (2) has vanishing bit error rate at any rate below  $C_{\text{iter}}$  (this follows by the standard density evolution analysis [31] using optimized degree sequences [12], [21]); and (3) has girth larger than  $k$  (therefore has low block error rate probability).

## II. DEFINITIONS AND PROBLEM STATEMENT

The *girth* of a graph  $G$  is defined to be the length of its shortest cycle. The degree of a node  $v$  of graph  $G$  is denoted by  $d_v(G)$ .

Consider two sequences of positive integers  $\bar{r} = r_1, \dots, r_n$  and  $\bar{c} = c_1, \dots, c_m$  for node-degrees such that  $e = \sum_{i=1}^n r_i = \sum_{j=1}^m c_j$ . We would like to generate a random bipartite graph  $G(V_1, V_2)$  with degree sequence  $(\bar{r}, \bar{c})$ , (i.e. for symbol nodes  $V_1 = \{u_1, \dots, u_n\}$ , check nodes  $V_2 = \{v_1, \dots, v_m\}$  we need  $\deg(u_i) = r_i$  and  $\deg(v_j) = c_j$ ) that has also girth greater than  $k$  (assume  $k$  is a constant and is an even number). We denote the set of all such graphs by  $\mathbb{G}_{\bar{r}, \bar{c}, k}$ .

A naïve approach would be to use *configuration* model to obtain a bipartite graph with degree sequence  $\bar{r}, \bar{c}$ . Let us quickly review the configuration model (for more details see [9]).

*Configuration model.* Let  $W^{(R)} = \cup_{i=1}^n W_i^{(R)}$  ( $W^{(C)} = \cup_{j=1}^m W_j^{(C)}$ ) be a set of  $e$  labeled mini-vertices with  $|W_i^{(R)}| = r_i$  ( $|W_j^{(C)}| = c_j$ ). Consider a procedure that finds a random perfect matching  $\mathcal{M}$  between mini-vertices of  $W^{(R)}$  and mini-vertices of  $W^{(C)}$  by choosing pairs of mini-vertices from  $W^{(R)} \times W^{(C)}$  sequentially and uniformly at random. Such a matching is also called a *configuration* on  $W^{(R)} \cup W^{(C)}$ . We can see that the number of all distinct configurations is equal to  $e!$ . Given a configuration  $\mathcal{M}$ , we can obtain a graph  $G_{\mathcal{M}}$  with degree sequence  $\bar{r}, \bar{c}$  by combining the mini-vertices of each  $W_i^{(R)}$  ( $W_j^{(C)}$ ) to form a vertex  $u_i$  ( $v_j$ ).

However, the resulting graph  $G_{\mathcal{M}}$  is in  $\mathbb{G}_{\bar{r}, \bar{c}, k}$  with very small probability which makes this approach impractical for generating codes with  $n, m$  very large.

Our sequential algorithm to sample uniformly from  $\mathbb{G}_{\bar{r}, \bar{c}, k}$  works as follows.

## Algorithm Bip-S

- (1) Set  $G_0$  to be a graph over vertex sets  $V_1 = \{u_1, \dots, u_n\}$ ,  $V_2 = \{v_1, \dots, v_m\}$  and with no edges. Let also  $\hat{r} = \{\hat{r}_1, \dots, \hat{r}_n\}$  and  $\hat{c} = \{\hat{c}_1, \dots, \hat{c}_m\}$  be two ordered set of integers that are initialized by  $\hat{r} = \bar{r}$  and  $\hat{c} = \bar{c}$ . For  $t = 0, \dots, e - 1$ , repeat the following steps:
  - If there is no *suitable* edges (any edge  $u_i v_j$  where the graph  $G_t \cup (u_i v_j)$  has no cycle of length at most  $k$ ), stop and return **FAIL**.
  - Consider the probability distribution  $p(u_i v_j | G_t)$  given by equation (1) below on the set of all suitable edges  $(u_i v_j)$ . Sample a suitable edge  $(u_i v_j)$  with distribution  $p(u_i v_j | G_t)$  and set  $G_{t+1} = G_t \cup (u_i v_j)$ .
  - Set  $\hat{r}_i = \hat{r}_i - 1$  and  $\hat{c}_j = \hat{c}_j - 1$ .
- (2) If the algorithm does not halt before  $t = e - 1$ , return  $G_e$ .

Here each probability  $p(u_i v_j | G_t)$  is an approximation to the probability that a uniformly random extension of graph  $G_t \cup (u_i v_j)$  has girth larger than  $k$  (see Section III for the intuition behind this). In order to find this approximation, we will consider a configuration model representation for the graphs with degree sequence  $(\bar{r}, \bar{c})$ . Then we use a similar argument to the one in Section 4 of [5], to find the following Poisson approximation for  $p(u_i v_j | G_t)$ :

$$p(u_i v_j | G_t) = \frac{\hat{r}_i \hat{c}_j e^{-E_k(G_t, u_i v_j)}}{Z(G_t)} \quad (1)$$

where  $Z(G_t)$  is a normalization constant, and  $\hat{r}_i, \hat{c}_j$ , denote the remaining degrees of  $i$  and  $j$ . Furthermore,

$$E_k(G_t, u_i, v_j) = \sum_{r=1}^{k/2} \sum_{\gamma \in \mathbb{C}_{2r}} \mathbb{I}_{\{(u_i v_j) \in \gamma\}} P_{ij}^t(\gamma) \quad (2)$$

where  $\mathbb{C}_{2r}$  is the set of all simple cycles of length  $2r$  in the complete bipartite graph on vertices of  $V_1$  and  $V_2$ , and  $P_{ij}^t(\gamma)$  is roughly the probability that  $\gamma$  is in a random extension of  $G_t$ . More precisely

$$P_{ij}^t(\gamma) = \frac{(e - t - 2r + |\gamma \cap G_t|)! \prod_{u_\ell \in \gamma} U_{ij}^t(u_\ell, \gamma) \prod_{v_\ell \in \gamma} V_{ij}^t(v_\ell, \gamma)}{(e - t - 1)!},$$

where

$$U_{ij}^t(u_\ell, \gamma) = \begin{cases} \hat{r}_\ell(\hat{r}_\ell - 1) & \text{If } d_{u_\ell}(\gamma \cap (G_t \cup (u_i v_j))) = 0, \\ \hat{r}_\ell & \text{If } d_{u_\ell}(\gamma \cap (G_t \cup (u_i v_j))) = 1, \\ 1 & \text{If } d_{u_\ell}(\gamma \cap (G_t \cup (u_i v_j))) = 2. \end{cases}$$

Similarly,

$$V_{ij}^t(v_\ell, \gamma) = \begin{cases} \hat{c}_\ell(\hat{c}_\ell - 1) & \text{If } d_{v_\ell}(\gamma \cap (G_t \cup (u_i v_j))) = 0, \\ \hat{c}_\ell & \text{If } d_{v_\ell}(\gamma \cap (G_t \cup (u_i v_j))) = 1, \\ 1 & \text{If } d_{v_\ell}(\gamma \cap (G_t \cup (u_i v_j))) = 2. \end{cases}$$

We explain the intuition and explanation of the above formula for  $p(u_i v_j | G_t)$  in Section III.

### III. THE INTUITION BEHIND ALGORITHM BIP-S

Define the *execution tree*  $T$  of the naïve algorithm described in Section II as follows. Consider a rooted  $m$ -level tree where the root (the vertex in level zero) corresponds to the empty graph in the beginning of the algorithm and level  $r$  vertices correspond to all couples  $(G_r, \pi_r)$  where  $G_r$  is a partial graph that can be constructed after  $r$  steps, and  $\pi_r$  is an ordering of its  $r$  edges. There is a link in  $T$  between a partial graph  $(G_r, \pi_r)$  from level  $r$  to a partial graph  $(G_{r+1}, \pi_{r+1})$  from level  $r+1$  if  $G_r \subset G_{r+1}$  and  $\pi_r, \pi_{r+1}$  coincide on the first  $r$  positions. Any path from the root to a leaf at level  $e$  of  $T$  corresponds to one possible way of generating a random graph in  $\mathbb{G}_{\bar{r}, \bar{c}, k}$ .

Let us denote those partial graphs  $G_r$  that have girth greater than  $k$  by “valid” graphs. Our goal is to reach a valid leaf in  $T$ , uniformly at random, by starting from the root and going down the tree. It is known that [34] in order to achieve this goal, at any step  $r$ , one needs to choose  $G_{r+1} = G_r \cup \{(ij)\}$  with probability proportional to the number of valid leaves of  $T$  that are descendant of  $(G_{r+1}, \pi_{r+1})$  (see [4] for a similar analysis in more details). Denote this probability by  $p(G_{r+1}, \pi_{r+1})$ .

The main technical contribution of this paper is deriving a new approximation  $\hat{p}(G_{r+1}, \pi_{r+1})$  for the true probabilities  $p(G_{r+1}, \pi_{r+1})$ , selecting  $(G_{r+1}, \pi_{r+1})$  with probability  $\hat{p}(G_{r+1}, \pi_{r+1})$ . For the case of general graphs with  $n$  vertices and  $m$  edges (no restriction on degrees) it was proven in [5] that the accumulated error  $\prod_{r=0}^{m-1} [\hat{p}(G_{r+1}, \pi_{r+1})/p(G_{r+1}, \pi_{r+1})]$  is small. Similar argument can be used here as well.

Consider a random variable  $n_k(G_{r+1}, \pi_{r+1})$  that is the number of cycles of length at most  $k$  in a leaf chosen uniformly at random from the descendants of  $(G_{r+1}, \pi_{r+1})$  in  $T$ . We can assume that the distribution of  $n_k(G_{r+1}, \pi_{r+1})$  behaves like Poisson<sup>2</sup>. That is the probability of  $n_k(G_{r+1}, \pi_{r+1}) = 0$  (i.e. reaching a valid leaf) is approximately  $\exp(-\mathbb{E}[n_k(G_{r+1}, \pi_{r+1})])$ . That explains the term  $e^{-E_k(G_t, u_i v_j)}$  in (1) and the first term  $\hat{r}_i \hat{c}_j$  in (1) shows the number of ways to match  $u_i$  and  $v_j$  in a configuration model.

In order to calculate  $E_k(G_t, u_i v_j)$ , due to additivity of the expectation we need to add the probabilities  $P_{ij}^t(\gamma)$  that each cycle  $\gamma$  of length  $2r$  ( $r = 2, \dots, k/2$ ), containing  $(u_i v_j)$ , is in a random extension of  $G_t \cup \{(u_i v_j)\}$  to a graph with degree sequence  $\bar{r}, \bar{c}$ .

The total number of configurations that lead to a random extension of  $G_t \cup \{(u_i v_j)\}$  is  $(e-t-1)!$ . On the other hand, For any fixed cycle  $\gamma$  with  $(u_i v_j) \in \gamma$ , there are exactly  $(e-t-2r+|\gamma \cap G_t|)! \prod_{u_\ell \in \gamma} U_{ij}^t(u_\ell, \gamma) \prod_{v_\ell \in \gamma} V_{ij}^t(v_\ell, \gamma)$  configurations that lead to a random extension which contains  $\gamma$ . The number  $U_{ij}^t(u_\ell, \gamma)$  ( $V_{ij}^t(v_\ell, \gamma)$ ) is the number of ways to choose the mini-nodes from  $W_\ell^{(R)}$  ( $W_\ell^{(C)}$ ). Hence,  $P_{ij}^t(\gamma)$  is the ratio of these two products.

<sup>2</sup>This can be shown using Janson’s inequality [2] (see [5] for a similar argument).

### IV. RUNNING TIME AND PRACTICAL IMPLEMENTATIONS

In this section we explain how to calculate the edge selection probabilities  $p(u_i v_j | G_t)$  in algorithm Bip-S. The dependence of the right-hand side of (1) on counting cycles, suggests the use of matrix multiplication. At each step  $t$  consider an  $4(e-t) + 2m + 2n$  by  $4(e-t) + 2m + 2n$  square matrix denoted by  $M_t = M_t(u_i v_j)$  that is constructed as follows. To simplify the notation let  $\hat{d} = \{\hat{d}_1, \dots, \hat{d}_{m+n}\}$  be the complete degree sequence  $\{\hat{r}, \hat{c}\}$ . That is  $\hat{d}_i = \hat{r}_i$  for  $1 \leq i \leq n$  and  $\hat{d}_i = \hat{c}_{i-n}$  for  $n+1 \leq i \leq n+m$ .

Consider  $M_t$  as an  $n+m$  by  $n+m$  matrix of blocks where its  $r, s$  block,  $M_t^{rs}$ , has size  $2\hat{d}_r + 2$  by  $2\hat{d}_s + 2$ .

- $M_t^{rs}$  is a 0-1 matrix. For  $r = s$ , all  $(a, b)$ -entries of  $M_t^{rs}$  are equal to zero except when  $a < \hat{d}_r + 1$  and  $b > \hat{d}_s + 2$ , or when  $a = \hat{d}_r + 1$  and  $b > \hat{d}_s + 1$ , or when  $a < \hat{d}_r + 2$  and  $b = \hat{d}_s + 2$  that the  $(a, b)$ -entries are equal to 1.
- If  $(u_r v_{s-n}) \in G_t \cup (u_i v_j)$  then all  $(a, b)$ -entries of  $M_t^{rs}$  are equal to zero except the entry  $a = \hat{d}_r + 2, b = \hat{d}_s + 1$  which is equal to 1. Similarly,  $M_t^{st}$  is an all zeros matrix except the entry with  $a = \hat{d}_s + 2, b = \hat{d}_r + 1$  which is equal to 1.
- If  $(u_r v_{s-n}) \notin G_t \cup (u_i v_j)$  then all  $(a, b)$ -entries of  $M_t^{rs}$  are equal to zero except the entries with  $a > \hat{d}_r + 2$  and  $b < \hat{d}_s + 1$  which are equal to  $\lambda_t = 1/(e-t)$ . Similarly,  $M_t^{sr}$  is all zeros except the entries with  $a > \hat{d}_s + 2$  and  $b < \hat{d}_r + 1$  which are equal to  $\lambda_t = 1/(e-t)$ .
- All blocks  $M_t^{rs}$  with both  $r$  and  $s$  less than  $n+1$  or both  $r$  and  $s$  greater than  $n$  is equal to zeros.

Now, if we approximate the factor  $(e-t-2r+|\gamma \cap G_t)!/(e-t-1)!$  by  $(e-t)^{-(2r-|\gamma \cap G_t|)}$ , it is not hard to see that  $\sum_{r=1}^{k/2} \sum_{\gamma \in \mathcal{C}_{2r}} \mathbb{I}_{\{(u_i v_j) \in \gamma\}} P_{ij}^t(\gamma)$  is roughly equal to the  $(a, b)$ -entry of the matrix

$$\frac{1}{2} \sum_{f=0}^{k/2} \left[ (M_t)^{4f+3} + ((M_t)^T)^{4f+3} \right]$$

where  $a = \sum_{r=1}^{i-1} (2\hat{d}_r + 2) + \hat{d}_i + 1$  and  $b = \sum_{s=1}^{n+j-1} (2\hat{d}_s + 2) + \hat{d}_j + 2$ .<sup>3</sup>

The fact that Algorithms Bip-S has polynomial running time is clear since the probabilities,  $p(u_i v_j | G_t)$ , at any step can be calculated using matrix multiplication. In fact a naïve calculation shows that  $p(u_i v_j | G_t)$  can be calculated with  $O(ke^3) = O(e^3)$  operations. This is because  $(M_t)^{4f+3}$  for any  $t, f$  takes  $O(fe^3)$  operations to compute. So we obtain the simple bound of  $O(e^4)$  for the running time of the algorithm Bip-S. But one can improve this running time by at least a factor  $e$  with exploiting the structure of the matrices. This is due to the fact the matrix  $M_t$  can be written as  $M_t = x_t^T y_t + S_t$  where  $x_t, y_t$  are  $4(e-t) + 2n + 2m$  by 1 vectors and  $S_t$  is a sparse matrix with  $O(e)$  entries. Thus, we obtain a bound of  $O(e^3)$  for the running time of Algorithm Bip-S. We leave a detailed analysis of the running time and implementation of the algorithms to the longer version of the paper.

<sup>3</sup>The use of transpose is because  $M_t$  is not a symmetric matrix and  $\sum_{f=0}^{k/2} (M_t)^{4f+3}$  only counts the directed paths.



Similar to [5], in order to check whether a pair  $(u_i v_j)$  is suitable or  $G_t \cup (u_i v_j)$  has no cycles of length at most  $k$ , we can use the adjacency matrix,  $A_t$ , of graph  $G_t$ . The pair  $(u_i v_j)$  is suitable if and only if the  $i, (j+n)$  entry of the matrix  $\sum_{r=0}^{k/2} (A_t)^{2r+1}$  is zero. This is equivalent to say that if and only if there is no path of length less than  $k$  between  $u_i$  and  $v_j$ . The same argument as in [5], shows that this can be done with  $O(e^2)$  operations at each iteration. Therefore, Algorithm Bip-S can be implemented with a total running time of  $O(e^3)$  which is  $O(n^3)$  when the graph has bounded degrees.

## V. DISCUSSION

We defer a more complete discussion of the codes generated by this algorithm to a complete version of the paper. Here we limit ourselves to a few remarks:

- (a) Several definitions have been proposed for the substructures responsible for the decoding errors at high signal-to-noise ratio. Our algorithm can be adapted to exclude any of these substructures (instead of cycles) as well.
- (b) In any of these definitions, the cycles play a dominant role. Therefore the above algorithm should be a good starting point.
- (c) In practical code design it can be preferable to partially structure the ensemble for facilitating the layout (as, for instance, in protograph codes [31]). Our graph generation procedure can be adapted to partially structured ensembles as well.

## REFERENCES

- [1] D. Alderson, J. Doyle, and W. Willinger, "Toward and Optimization-Driven Framework for Designing and Generating Realistic Internet Topologies", *HotNets*, 2002.
- [2] N. Alon and J. Spencer, "The Probabilistic Method", *Wiley, NY*, 1992.
- [3] A. Amraoui, A. Montanari and R. Urbanke, "How to Find Good Finite-Length Codes: From Art Towards Science", *Eur. Trans. Telecomm.* 18 (2007), 491-508
- [4] M. Bayati, J. H. Kim and A. Saberi, "A Sequential Algorithm for Generating Random Graphs", *International Workshop on Randomization and Computation*, 2007.
- [5] M. Bayati, A. Montanari and A. Saberi, "Generating Random Graphs with Large Girth", *SODA*, 2009.
- [6] J. Blanchet, "Efficient Importance Sampling for Binary Contingency Tables", preprint, 2007.
- [7] J. Blitzstein and P. Diaconis, "A sequential importance sampling algorithm for generating random graphs with prescribed degrees", preprint, 2006.
- [8] B. Bollobás, and O. Riordan, "Constrained graph processes", *Electronic Journal of Combinatorics*, 7, 2000.
- [9] B. Bollobás, *Random Graphs*, Cambridge University Press, 2001.
- [10] T. Bu and D. Towsley, "On Distinguishing between Internet Power Law Topology Generator", *INFOCOM*, 2002.
- [11] Y. Chen, P. Diaconis, S. Holmes and J.S. Liu, "Sequential Monte Carlo methods for statistical analysis of tables", *Journal of the American Statistical Association* 100, 109-120, 2005.
- [12] S. Y. Chung, G. D. Forney, Jr., T. Richardson and R. Urbanke, "On the design of low-density parity-check codes within 0.0045 dB of the Shannon limit," *IEEE Comm. Lett.* 5 (2001), 58-60
- [13] M. Faloutsos, P. Faloutsos and C. Faloutsos, "On Power-law Relationships of the Internet Topology", *SIGCOM*, 1999.
- [14] X. Hu, E. Eleftheriou and D. Arnold, "Progressive Edge-Growth Tanner Graphs", *Proc. of GLOBECOM 01*, San Antonio, USA, Nov 2001
- [15] X. Hu, E. Eleftheriou and D. Arnold, "Regular and irregular progressive edge-growth Tanner graphs", *IEEE Trans. on Inform. Theory*, 51 (2005), 386-398
- [16] P. Erdős, S. Suen, and P. Winkler, "On the size of a random maximal graph", *Random Structure and Algorithms*, 6, pp 309-318, 1995.
- [17] S. Janson, T. Łuczak, and A. Ruciński, "Random Graphs", *Wiley-Interscience*, 2000.
- [18] J. H. Kim, V. H. Vu, "Generating random regular graphs", *ACM Symposium on Theory of Computing (STOC)*, 213-222, 2003.
- [19] D. Knuth, "Mathematics and computer science: coping with finiteness", *Science* 194(4271):1235-1242, 1976.
- [20] R. Koetter and P. Vontobel, "Graph covers and iterative decoding of finite-length codes," *Int. Conf. on Turbo codes and Rel. Topics*, Brest, France, September 2003.
- [21] M. Luby, M. Mitzenmacher, A. Shokrollahi, D. A. Spielman and V. Stemann, "Practical Loss-Resilient Codes," 29th annual ACM Symposium on Theory of Computing (STOC), 1997, pp. 150-159
- [22] D. J. MacKay and M. S. Postol, "Weaknesses of Margulis and Ramanujan-Margulis Low-Density Parity-Check Codes", *Elec. Notes in Theor. Comp. Science* 74 (2003)
- [23] A. Medina, I. Matta and J. Byers, "On the origin of power laws in Internet topologies", *ACM Computer Communication Review*, vol. 30, no. 2, pp. 18-28, 2000.
- [24] R. Milo, S. ShenOrr, S. Itzkovitz, N. Kashtan, D. Chklovskii and U. Alon, "Network motifs: Simple building blocks of complex networks", *Science* 298, 824-827, 2002.
- [25] A. Montanari, "The asymptotic error floor of LDPC ensembles under BP decoding" 44rd Allerton Conference on Comm. Control and Comp., Monticello, IL, Sep 2006
- [26] J. M. F. Moura and J. Lu and H. Zhang "Structured low-density parity-check codes", *IEEE Sign. Proc. Mag.*, 21 (2004), 42-55
- [27] D. Osthus, and A. Taraz, "Random maximal H-free graphs", *Random Structures and Algorithms*, 1999.
- [28] C. Di, D. Proietti, I. E. Teletar, T. J. Richardson and R. Urbanke, "Finite-length analysis of low-density parity-check codes on the binary erasure channel", *IEEE Trans. Inform. Theory*, 46 (2002) 1570-1579
- [29] A. Ramamoorthy and R. Wesel, "Construction of short block length irregular low-density parity-check codes" *IEEE Intern. Conf. on Comm.*, Paris, June 2004
- [30] T. Richardson, "Error floors of LDPC codes", 41st Allerton Conference on Comm. Control and Comp., Monticello, IL, Sep 2003
- [31] T. Richardson and R. Urbanke, *Modern Coding Theory*, Cambridge University Press, Cambridge 2008
- [32] J. Rosenthal and P. O. Vontobel, "Constructions of regular and irregular LDPC codes using Ramanujan graphs and ideas from Margulis", *IEEE Symp. on Inform. Theory*, Washington, June 2001
- [33] A. Ruciński, and N. Wormald, "Random graph processes with degree restrictions", *Combin. Prob. Comput.* 1, pp 169-180, 1992.
- [34] A. Sinclair, "Algorithms for random generation and counting : a Markov chain approach", *Birkhäuser*, 1993.
- [35] A. Steger and N.C. Wormald, "Generating random regular graphs quickly", *Combinatorics, Probab. and Comput* (8), pp 377-396, 1999.
- [36] A. Steger, "On the Evolution of Triangle-Free Graphs", *Combinatorics, Probability and Computing*, 14, pp 211-224, 2005.
- [37] J. Spencer, "Counting extensions", *J. Combin. Theory Ser. A* 53, pp247-255, 1990.
- [38] J. Spencer, "Maximal triangle-free graphs and Ramsey R(3,t)", unpublished manuscript, 1995.
- [39] M. G. Stepanov, V. Y. Chernyak and M. Chertkov, "Diagnosis of weaknesses in modern error correction codes: a physics approach", *Phys Rev. Lett.* 95 (2005) 228701-228704
- [40] H. Tangmunarunkit, R. Govindan, S. Jamin, S. Shenker, and W. Willinger, "Network Topology Generators: Degree based vs. Structural", *ACM SIGCOM*, 2002.
- [41] N. C. Wormald, "Models of random regular graphs", *Surveys in combinatorics (Canterbury)* London Math. Soc. Lecture Notes Ser., Vol 265. Cambridge Univ. Press, Cambridge, 239-298, 1999.
- [42] H. Xiao and A. H. Banihashemi, "Improved progressive-edge-growth (PEG) construction of LDPC codes", *Proc. of GLOBECOM 04*, Dallas, USA, Nov 2004