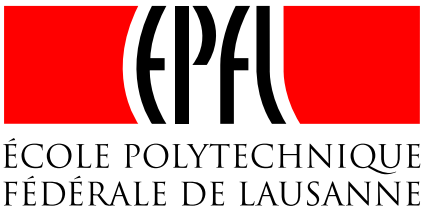

A SOURCE CODING SCHEME USING SPARSE GRAPHS

RETHNAKARAN PULIKKOONATTU

ABSTRACT. This is the report that I have submitted towards the final examination evaluation of the doctoral course *Modern Coding theory* by Ruediger Urbanke at EPFL during Spring 2008. The problem studied can be roughly stated as source coding or compression using sparse graphs. We discuss a simple source coding problem and analyze its performance using an analytical tool formulated by *Wormald*, usually employed to describe the expected behaviour of a suitably conditioned stochastic process on graph. A simulation investigation of the chosen scheme is also presented to verify the theoretical analysis.

Rethnakaran Pulikkoonattu
Professor: Rüdiger Urbanke

Date: 2008 July 10.
Key words and phrases. LDGM,Wormald,Modern Coding theory, Tanner graph.
Laboratoire de théorie des communications, Information processing Group, EPFL, Lausanne, Switzerland,1005, ipg.epfl.ch.

1. Examination problem, Modern Coding theory Spring 2008

We want to compress a binary symmetric source (BSS) at a rate R using iterative techniques. [If you are not familiar with the lossy source compression problem, please stop by so we can discuss it, or consult the book by Cover and Thomas.] A BSS emits iid Bernoulli(1 2) random variables. It is well known that the smallest Hamming distortion which we can achieve with any scheme is $D_{\text{Shannon}} = h_2^{-1}(1-R), 0 \leq R \leq 1$, or in other words, that the smallest rate which we need to achieve Hamming distortion $D, 0 \leq D \leq \frac{1}{2}$, is $D_{\text{Shannon}} = 1 - h_2(D)$, where $h_2()$ is the binary entropy function. Our aim is to analyze and simulate the following very simple iterative scheme. Slightly more sophisticated versions of this scheme are known to be able to achieve a performance very close to the rate-distortion function. We use an LDGM code as discussed in class (see the course notes), with m information bits and n code bits. Let C denote the set of codewords of the LDGM code. Assume we are given a source word s of length n. (Note: there are source bits and information bits, these are not the same.) The encoding task consists of finding a codeword $x \in C$ which has minimum Hamming distortion with s:

$$\hat{x}(s) = \arg\min_{x \in C} \quad d(x, s) \tag{1}$$

where $d(\star, \star)$ is the Hamming distance. Let u be the information word which generates x. Since $m < n$, we then have compressed the source word s to the smaller information word u. The incurred distortion is $d(x(u), s)$. In general it is difficult to find the best representative x for a given source word s. We will therefore attempt to find a good representative in a greedy fashion. Let us first define the specific LDGM ensemble which we consider. All variables have degree 3. The check nodes have a Poisson distribution. More precisely, for each variable node we pick 3 check nodes uniformly at random.

In order to find a good representative we proceed as follows. We are given the source word s. In the sequel we will say that check node $i, i \in [n]$, has value $\alpha, \alpha \in \{0, 1\}$, to mean that $s_i = \alpha$. We want to find the values of the information word u, so that the induced codeword $x = x(u)$ has a small distortion. We proceed in a greedy fashion by fixing the value of one information node at a time. At any point in time some of the information bits have been set and some other ones are still free. Once we set the value of an information node we delete its edges from the graph. This gives us a sequence of residual graphs. To determine at time t which information bit to set and what value to set it to we proceed as follows. For each yet undecided information node we determine its type. The type of an information node is the number of connected check nodes of residual degree 1. The type is therefore an integer in the range $0, 1, 2, 3$. Let $N_i(t)$ denote the number of variables in the residual graph at time t of type i. Let $w_i, i \in 0, 1, 2, 3$, denote weights, i.e., non-negative reals. The algorithm picks an information bit of type i with probability $\frac{w_i N_i(t)}{\sum_{j=0}^{3} w_j N_j(t)}$

It then sets the value of the chosen information bit to the majority of the values of all connected check nodes that have residual degree 1. (If there is no clear majority the algorithm picks the value uniformly at random.) The algorithm adds the chosen value to all connected check nodes and then removes all the edges adjacent to the chosen variable. After m steps the algorithm stops. We are interested in the resulting distortion of this scheme for given weights w_i as well as how we should choose these weights in order to minimize the distortion. You are asked to analyze this scheme as well as to implement it and to show that your analysis matches the simulation results. How does your result compare to the Shannon rate-distortion bound? Once you managed the simplest case there are many possible extensions you can try. Can you think of ways to improve upon the given scheme in order to achieve a better rate-distortion performance? What is the best scheme you can think of and how well does it perform?

2. Source coding problem

The rate distortion for Bernoulli source [3] is given by,

$$R(D) = 1 - h_2(D), D \in \left[0, \frac{1}{2}\right] \tag{2}$$

In essence, rate distortion theorem states that, for a given rate (compression rate) R, there is certain guaranteed minimum distortion, however best the algorithm is. Compression aka source coding problem with certain admissible (non zero) distortion refers to noisy compression. In this report, we discuss an approach to lossy coding using iterative schemes on sparse graphs[1]. More precisely using Low density parity generator matrix (LDGM) code. Sparse graphs based source coding is addressed in many papers such as [7], [9],[10].

3. Factor graph representation of the LDGM code

We will briefly discuss the bipartite representation of an LDGM code. The example presented here is taken from [8].

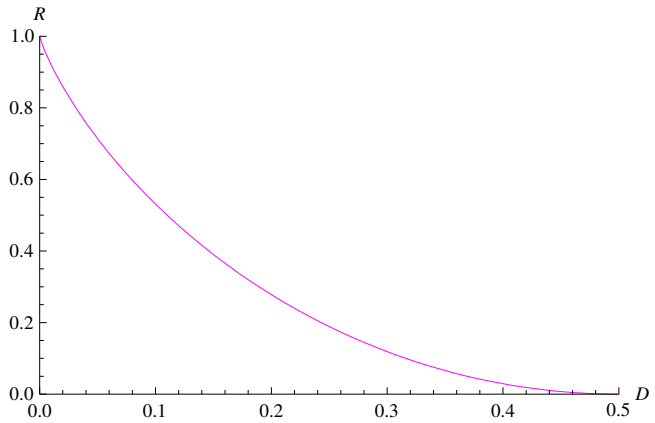

FIGURE 1. Rate distortion curve for binary Bernoulli source. The region north-east (up-right) to the curve is the achievable region.

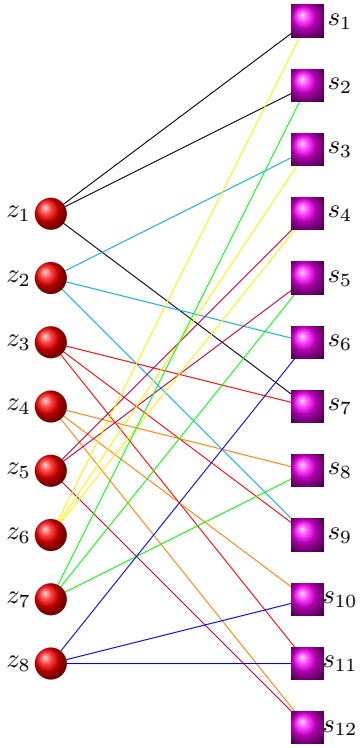

FIGURE 2. Factor graph representation of the LDGM code. The check nodes correspond to the source bits $s_1, s_2, \ldots, s_n$. The information word is given by $z_0, z_1, \ldots, z_m$ where $m = nR$. In this example, $n = 2$ and $m = 8$. The variable node degree is fixed and equal to 3 as prescribed in the problem statement. In this example, $d_c = 2$ is fixed, but for the simulation and the model we considered (in this report), the degree d_v is not fixed to any value. Only $d_v = 3$ is fixed. In other words, the degree of the check nodes can be any value and is random.

The check nodes correspond to the source bits $s_1, s_2, \ldots, s_n$. The information word is given by $z_0, z_1, \ldots, z_m$ where $m = nR$. In the factor graph representation shown in Figure.2, $n = 2$ and $m = 8$. The variable node degree is fixed and equal to 3 as prescribed in the problem statement[1].

A bipartite graph of a simple regular graph is shown in Figure.2. All the check node have degree 2 and variable node have degree 3.

3.1. **Algorithm.** The source coding algorithm is as follows[1]. Algorithm has a finite number of iterations. The number of iterations equals the number of variable nodes. Every step, one variable node is eliminated. At every stage, a variable node is picked based on its prevailing distribution. The majority value of degree one check connected to the chosen variable node, is assigned to the variable node. Once the value is assigned, the variable node as well as (all) the connected edges are removed. The graph evolves to a new graph with one less variable node. In the end, all variable nodes and check nodes are disconnected and there the algorithm stop.

4. Theoretical Analysis

Analysis of this algorithm is performed under the framework of a stochastic process on graph.

4.1. **Wormald method.** The algorithm described can be brought into the framework of a stochastic process evolution on Tanner graph and hence we can bring in the Wormald method [2, 4, 5, 6] for analyzing the expected decoding performance. Wormald method serve as an analytical tool to study the expected behavior of stochastic processes. It is based on the idea that, a system after a series of random steps, eventually with very high probability, will stay close to the expected behavior [4, 5, 6]. This behavior can further be determined by a set of differential equations.

Let us consider a graph random process $G(t)$. The process starts with graph $G(0)$ from which edges are repeatedly removed according to a probabilistic rule, that is a priori known. This removal procedure results in a probabilistic set of sequences $G(0), G(1), \ldots, G(t)$, where $G(t)$ denotes the t-th graph in that process. When no edge can be removed from $G(t)$ (subject to the removal constraints) the process becomes stationary. Thus, we have $G(t+1) = G(t)$ and this is the final graph of this stochastic process. Usually this happens for large t ($t \to \infty$). However, for many systems the above procedure terminates after finite number of steps (time $t = T_d$), forming a family of graphs $G(0), G(1), \ldots, G(T_d)$. Wormald method studies the behavior of such sequence of processes.

Since the employed encoding scheme is described by a stochastic process, Wormald method can be used for the analysis of its performance. The residual Tanner graph $G(t)$ is characterized by a set of pairs $(V_i(t), C_i(t)), \forall i$ where $V_i(t)$ and $C_i(t)$ denote respectively the total number of edges connected with variable nodes and check nodes of degree i at time t.

4.2. **Wormald theorem.** In this section, we apply Wormald theorem to analyze the source coding problem we are investigating. A formal proof of the applicability of the Wormald theorem on the decoding algorithm for a Peeling Decoder can be found in [2]. We present here a general framework on the Wormald theorem and its application to the source coding problem under consideration is discussed in subsequent section.

Let us consider now that $G(t)$ has a state space $\{0, \ldots, \theta\}^d, d \in \mathbb{N}$ and a probability space $\mathcal{S}$. Consider a sequence $\{G^m(t)\}_{m>1}$ of a Markov random process where $G_i^m(t)$ is the i-th component of $G^m(t)$. Denote a subset $\Gamma \subset \mathbb{R}^{d+1}$ containing the vectors $[0, g_1, \ldots, g_d]$ such that,

$$\mathbb{P}\left(\frac{G_i^{(m)}(t=0)}{m} = g_i\right) > 0, \forall 1 \leq i \leq d, m > 1 \tag{3}$$

Let f_i be functions from $\mathbb{R}^{d+1}$ to $\mathbb{R}^d$, satisfying the following conditions.

(1) For $t < m$, there exists a constant c_i^m such that,

$$\left|G_i^{(m)}(t+1) - G_i^{(m)}(t)\right| \leq c_i^m, 1 \leq i \leq d \tag{4}$$

(2) For $t < m$ and $\forall 1 \leq i \leq d$,

$$\begin{aligned} &\mathbb{E}\left[G_i^{(m)}(t+1) - G_i^{(m)}(t) | G^{(m)}(t)\right] \\ &\triangleq f_i\left(\frac{t}{m}, \frac{G_1^{(m)}(t)}{m} \ldots, \frac{G_d^{(m)}(t)}{m}\right) \end{aligned} \tag{5}$$

(3) $f_i, \forall i \leq d$ is Lipschitz continuous function on the intersection of Γ with the half space $\{(t, g_1, \ldots, g_d) : t \geq 0\}$, *i.e.*, if $x, y \in \mathbb{R}^{d+1}$ belong to this intersection, then there exists a Lipschitz constant ζ such that,

$$|f_i(x) - f_i(y)| \leq \zeta \sum_{i=1}^{d+1} |x_i - y_i| \tag{6}$$

Under these above conditions, the following holds true:

(1) For the vector $[t, g_1, g_2, \ldots, g_d] \in \Gamma$, the system of differential equations

$$\frac{\partial g_i}{\partial \tau} = f_i\left(\tau, g_1, \ldots, g_d\right), 0 \leq i \leq d \tag{7}$$

has a unique solution for $g_i(\tau) : \mathbb{R} \to \mathbb{R}$ in Γ with the initial condition $g_i(0) = x_i, 1 \leq i \leq d$.

(2) There exist a strictly positive constant δ such that

$$\mathbb{P}\left(\left|\frac{G_i^{(m)}(t)}{m} - g_i\left(\frac{t}{m}\right)\right| \geq \delta m^{-\frac{1}{6}}\right) < \frac{dm^{\frac{2}{3}}}{e^{\frac{m^{\frac{1}{3}}}{2}}} \tag{8}$$

for $0 \leq t \leq m\tau_{\max}$ and for each i. The term g_i is the unique solution obtained by solving Eq. (7) with the initial conditions $g_i(0) = \mathbb{E}\left[\frac{G_i^{(m)}(t=0)}{m}\right]$ and $\tau_{\max} = \tau_m$ is the supremum of those τ to which the solution can be extended, under some boundedness criteria [4, 5, 6, 2].

In other words, Eq. (8) states that, when m is big enough, each realization of the process $G_i^{(m)}(t)$ is close to the (unique) solution of Eq. (7), with high probability.

4.3. **Analysis of the source encoding scheme using Wormald method.** The analysis of the source coding problem using Wormald method is very similar to the preferential attachment algorithm dealt in [11]. We use the following notations. The source word length (length message to be compressed) is n and the length of information word (compressed word length) is m. The rate of compression R is related by $m = nR$, where $R \in [0, 1]$. In the factor graph (essentially a bipartite representation) shown in figure, the check nodes represent the source word (n elements of the source encoder input) and variable nodes (m elements) are the source encoder output.

At start, all variable nodes have degree 3. The edges are connected to check nodes uniformly at random, which means, the check degree follows a Poisson distribution. So, at start, the check node degrees follow a certain distribution (Poisson). Eventually, when the algorithm stops (after $m = nR$ time steps), all edges are removed. In other words, all check nodes attain degree 0.

For analysis, let us assume, the maximum degree of a check node (at time $t = 0$) is $d_c^{\max}$. The graph evolves with time (time step) and let $G(t)$ be the residual graph after the time step t. The check node degree distribution change with time t. We denote $c_j(t)$ for the number of edges in $G(t)$ connected to check nodes of degree j, where $i = 0, 1, 2, \ldots, d_c^{\max}$. The graph evolves from time $t = 0$ to $t = m$ (After that, it remains so).

We first study the expected behaviour of the check node degree distribution. For large n, we derive a set of differential equations corresponding to this behaviour. We assume the expected behaviour gets close to the individual behaviour. Solving the differential equations will give us the degree distribution of the check nodes.

The algorithm enforces certain distortion (some cases none as well), at every time step. More precisely, when the picked variable nodes have degree 1, all such degree 1 nodes are disconnected from the graph (no edges) and thus the residual distortion caused by these detached check nodes stay on. Let us denote this instantaneous residual distortion (slight abuse of naming perhaps!) by $\Delta D(t)$. Then $D(t)$, the distortion at time t is simply the accumulated value of these residual distortions and is equal to $\sum_{t=0}^{t} \Delta D(t)$.

To compute the $\Delta D(t)$ residual distortion at t, we need to know the probability of the variable nodes with type k, which in turn can be computed from the values of $c_j(t), j = 0, 1, \ldots d_c^{\max}$.

(1) The probability that a chosen variable node (at time step t) has type $k = 2$ is equal to

$$P_2 = \binom{3}{2} \frac{c_1(t)}{\sum_{k\geq 1} c_k(t)} \frac{c_1(t) - 1}{\sum_{k\geq 1} c_k(t)} \frac{\sum_{k\geq 2} c_k(t)}{\sum_{k\geq 1} c_k(t)}$$

The average residual distortion due to this is $\frac{1}{2n}$ (Two check nodes settle certain distortions in this step. 00 and 11 values result no distortion whereas 10 or 01 would cause $1/n$. Average hence, is $\frac{2}{4n}$).

(2) The probability that a chosen variable node (at time step t) has type $k = 3$ is equal to

$$P_3 = \binom{3}{3} \frac{c_1(t)}{\sum_{k\geq 1} c_k(t)} \frac{c_1(t) - 1}{\sum_{k\geq 1} c_k(t)} \frac{c_1(t) - 2}{\sum_{k\geq 1} c_k(t)}$$

The distortion measure is Hamming distortion[8]. The average residual distortion due to this is $\frac{6}{8n}$ (Three check nodes settle their distortions in this step. 000 and 111 values result no distortion whereas the other 6 possibilities build $1/n$ apiece. Average hence, is $\frac{6}{8n}$).

(3) Type 0 nodes cause no distortion (They settle to zero distortion).

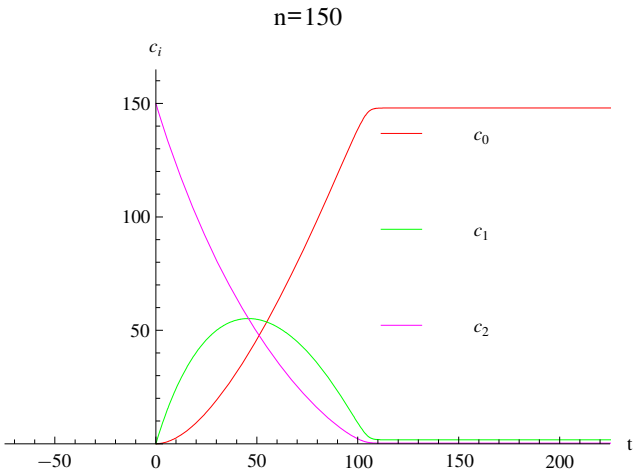

FIGURE 3. Evolution of the degree distribution of edges: The number of edges connected to check nodes of degree $i, i = 0, 1, 2$ is are shown, as it evolves with time. The x-axis is the time step. At start (i.e., $t = 0$), all the edges are connected to check nodes of degree 2. This refers to $c_0 = 0 = c_1 = 0, c_2 = n$. After m steps, all the edges are peeled off (edges are connected to none of the check nodes), which is equivalent to $c_0 = n$. The evolution is captured for $n = 150, m = 100$.

So, the average residual distortion is then

$$\Delta D(t) = \frac{6}{8n} P_3(t) + \frac{1}{2n} P_2(t) \tag{9}$$

We require $c_j(t)$ to compute the distortion. For that, we can use Wormald method, where we solve a system of differential equations as discussed in [2]. In general, we require $d_c^{\max}$ number of differential equations and solve them. Here, we first consider a small example to ease up the computations. Later we consider a more general setup with Poisson distributed check node degree distribution. The principle and machinery are similar. We discuss the simple, regular (not Poisson) case next.

4.4. **Analysis of a** $(2, 3)$ **regular graph.** We consider the case $d_c^{\max} = 2$. We deal this case in the next subsection. Let us consider a simpler case with the check nodes all have degree equal to 2. This will simplify the analysis a lot. We use Wormald method to compute the expected change in the degree distribution (from an edge perspective).

$$\begin{aligned}
\frac{\partial c_0}{\partial t} &= 3\frac{c_1(t)\,(c_1(t)-1)\,(c_1(t)-2)}{(c_1(t)+c_2(t))^3} + 2\binom{3}{1}\frac{c_1(t)\,(c_1(t)-1)\,c_2(t)}{(c_1(t)+c_2(t))^3} + 1\binom{3}{1}\frac{c_1(t)\,(c_2(t)-1)\,c_2(t)}{(c_1(t)+c_2(t))^3} \\
\frac{\partial c_1}{\partial t} &= 3\frac{c_2(t)\,(c_2(t)-1)\,(c_1(t)-2)}{(c_1(t)+c_2(t))^3} + \binom{3}{2}\frac{c_2(t)\,(c_2(t)-1)\,c_1(t)}{(c_1(t)+c_2(t))^3} - \\
&\quad \binom{3}{2}\frac{c_1(t)\,(c_1(t)-1)\,c_2(t)}{(c_1(t)+c_2(t))^3} - 3\frac{c_1(t)\,(c_1(t)-1)\,(c_2(t)-2)}{(c_1(t)+c_2(t))^3} \\
\frac{\partial c_2}{\partial t} &= -3\frac{c_2(t)c_1(t)\,(c_1(t)-1)}{(c_1(t)+c_2(t))^3} - 2\binom{3}{2}\frac{c_2(t)\,(c_2(t)-1)\,c_1(t)}{(c_1(t)+c_2(t))^3} - 3\binom{3}{3}\frac{c_2(t)\,(c_2(t)-1)\,(c_2(t)-2)}{(c_1(t)+c_2(t))^3}
\end{aligned}$$

We have used Mathematica to solve these equations numerically[12].

4.5. **Analysis with Poisson distribution.** Now, we can use the same methodology to extend the case of irregular check node degree distribution. More specifically, we analyse a Poisson distributed check node (degree) configuration. The distribution values are chosen from a histogram obtained from a random realization of the graph $G(t)$ at $t = 0$. Here, $d_c^{\max} = 10$. The Poisson histogram of a graph is shown in Figure.6.

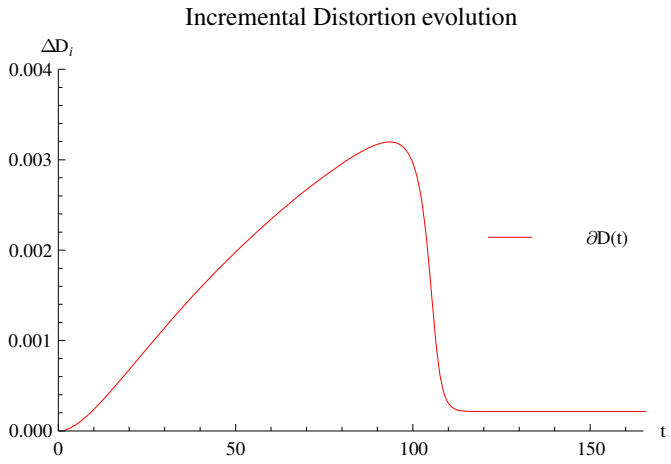

FIGURE 4. Regular graph $(2,3)$. The incremental distortion evolution with time. At every time step, a certain irrecoverable distortion creep in. The eventual distortion at any time is then the accumulated value of this. An integral of this curve should correspond to the average instantaneous distortion. Since algorithm stops after m steps, the average distortion (overall) is then the accumulated value at $t = m$.

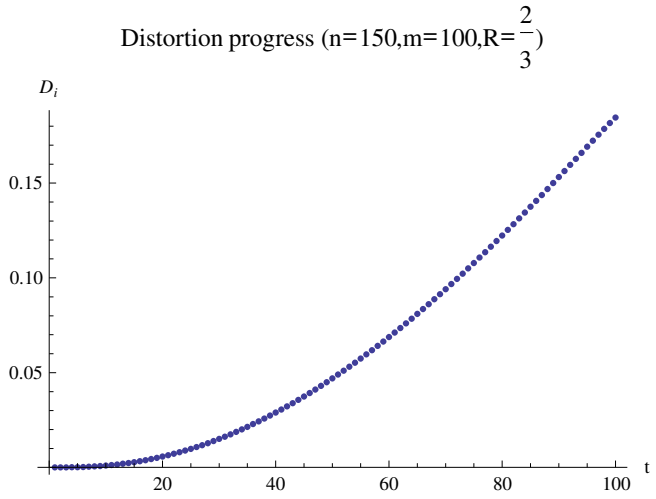

FIGURE 5. Regular graph $(2,3)$.The distortion evolution with time. Figure shows the average distortion evolved as time progresses. The eventual distortion is the value at $t = m$. In this example, for $2,3$ regular configuration (rate $R = 2/3$), the distortion progress is shown. This curve is essentially the integrated values of the residual distortion of Fig.4. The average distortion for rate $R = 2/3$ is approximately 0.18 which closely match the simulation result. The figure correspond to $m = 100, n = 150$.

The differential equations for the degree evolution of the graph is given below. For $1 \leq i \leq d_c^{\max} - 1$,

$$\begin{aligned}
\frac{\partial c_i}{\partial t} &= -3\frac{c_i(t)\left(\sum_{j\geq 1, j\neq, i+1}^{d_c^{\max}} c_j(t)\right)\left(\sum_{j\geq, j\neq i, i+1}^{d_c^{\max}} c_j(t)\right)}{\left(\sum_{j=1}^{d_c^{\max}} c_j(t)\right)^3} + 3\frac{c_i(t)(c_{i+1}(t))\,(c_{i+1}(t)-1)}{\left(\sum_{j=1}^{d_c^{\max}} c_j(t)\right)^3} \\
&\quad - 6\frac{c_i(t)(c_i(t)-1)\left(\sum_{j\geq, j\neq i, i+1}^{d_c^{\max}} c_j(t)\right)}{\left(\sum_{j=1}^{d_c^{\max}} c_j(t)\right)^3} - 3\frac{c_i(t)(c_i(t)-1)\,(c_{i+1}(t))}{\left(\sum_{j=1}^{d_c^{\max}} c_j(t)\right)^3} - 3\frac{c_i(t)(c_i(t)-1)\,(c_i(t)-2)}{\left(\sum_{j=1}^{d_c^{\max}} c_j(t)\right)^3} \\
&\quad + 3\frac{c_{i+1}(t)(c_{i+1}(t)-1)\,(c_{i+1}(t)-2)}{\left(\sum_{j=1}^{d_c^{\max}} c_j(t)\right)^3} + 3\frac{c_{i+1}(t)\left(\sum_{j\geq 1, j\neq, i+1}^{d_c^{\max}} c_j(t)\right)\left(\sum_{j\geq, j\neq i, i+1}^{d_c^{\max}} c_j(t)\right)}{\left(\sum_{j=1}^{d_c^{\max}} c_j(t)\right)^3} \\
&\quad + 6\frac{c_{i+1}(t)(c_{i+1}(t)-1)\left(\sum_{j\geq, j\neq i, i+1}^{d_c^{\max}} c_j(t)\right)}{\left(\sum_{j=1}^{d_c^{\max}} c_j(t)\right)^3}
\end{aligned}$$

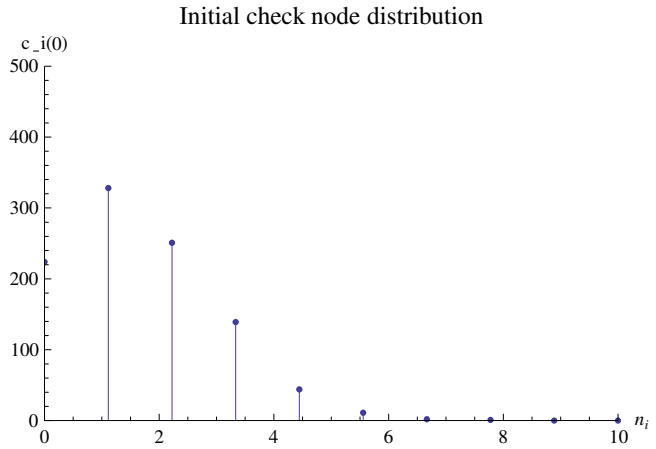

FIGURE 6. Poisson distribution (histogram of a sample graph with $n = 1000$ check nodes) of a graph.

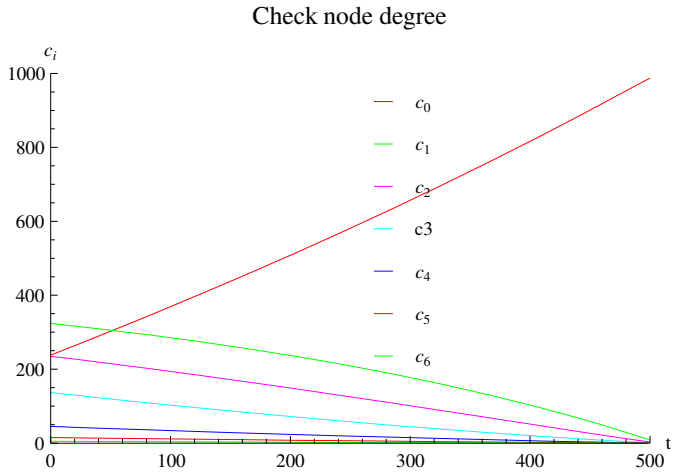

FIGURE 7. Evolution of the degree distribution of edges: The number of edges connected to check nodes of degree $i, i = 0, 1, 2$ is are shown, as it evolves with time. The x-axis is the time step. At start (i.e., $t = 0$), the edges are connected to check nodes of degree drawn from Poisson distribution. This refers to Poisson distributed c_i. After m steps, all the edges are peeled off (edges are connected to none of the check nodes), which is equivalent to $c_0 = n$. The evolution is captured for $n = 1000, m = 500$ ($R = 1/2$).

For $i = 0$, we have,

$$\frac{\partial c_0}{\partial t} = 3\frac{c_1(t)(c_1(t)-1)\,(c_1(t)-2)}{\left(\sum_{j=1}^{d_c^{\max}} c_j(t)\right)^3} + 6\frac{c_1(t)(c_1(t)-1)\left(\sum_{j\geq 2}^{d_c^{\max}} c_j(t)\right)}{\left(\sum_{j=1}^{d_c^{\max}} c_j(t)\right)^3} + 3\frac{c_1(t)\left(\sum_{j\geq 2}^{d_c^{\max}} c_j(t)\right)\left(\sum_{j\geq 2}^{d_c^{\max}} c_j(t)-1\right)}{\left(\sum_{j=1}^{d_c^{\max}} c_j(t)\right)^3}$$

For $i = d_{\max}$, we have,

$$\frac{\partial c_i}{\partial t} = -3\frac{c_i(t)(c_i(t)-1)\,(c_i(t)-2)}{\left(\sum_{j=1}^{d_c^{\max}} c_j(t)\right)^3} - 6\frac{c_i(t)(c_i(t)-1)\left(\sum_{j\geq 1}^{d_c^{\max}-1} c_j(t)\right)}{\left(\sum_{j=1}^{d_c^{\max}} c_j(t)\right)^3} - 3\frac{c_i(t)\left(\sum_{j\geq 1}^{d_c^{\max}-1} c_j(t)\right)\left(\sum_{j\geq 1}^{d_c^{\max}-1} c_j(t)-1\right)}{\left(\sum_{j=1}^{d_c^{\max}} c_j(t)\right)^3}$$

The evolution of the check node degree distribution as it progresses with time is captured in Figure.7.

The differential equations are solved numerically, using mathematica[12]. All the relevant figures on analysis are thus obtained using the Mathematica script.

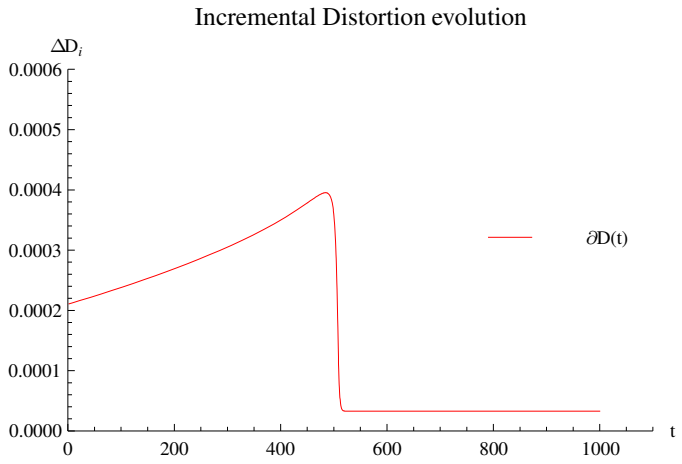

FIGURE 8. Instantaneous average additional distortion at every time evolution. The distortion accumulates over time and when the algorithm stops after m steps, the cumulative distortion is the average distortion. Result shown here correspond to the configuration with $n = 1000$ and $m = 500$ ($R = 1/2$). The variable nodes all have degree 3 whereas, the check node degrees are drawn from Poisson distribution. The instantaneous (irrecoverable) distortion progress with each iteration until the last time step $t = m$. The average distortion is the accumulated value of these instantaneous distortions at time $t = m$.

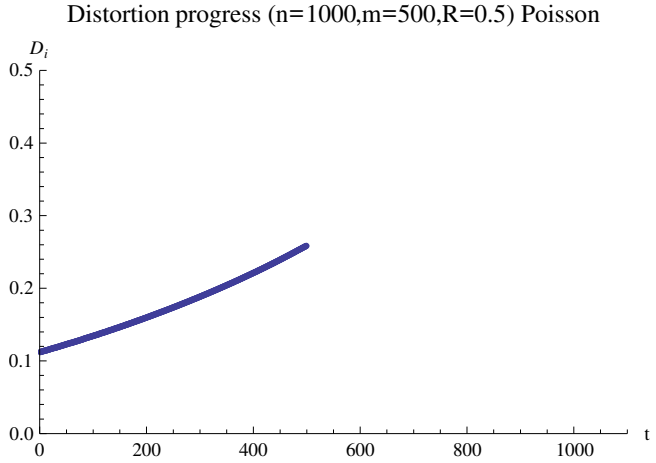

FIGURE 9. Average distortion evolve with time.Result shown here correspond to the configuration with $n = 1000$ and $m = 500$ ($R = 1/2$). The variable nodes all have degree 3 whereas, the check node degrees are drawn from Poisson distribution. The figure is obtained as accumulated trace of Figure.8. The average distortion is simply the value at $t = m$.

5. SIMULATION OF LDGM CODE FOR SOURCE CODING

The evolution of the degrees have behaviour matching with the analysis. The evolution for the regular graph $(2, 3)$ is shown in Figure.10 and that of Poisson distributed check nodes is shown in Figure.11. We can observe that, Figure.10 has the same evolution as predicted by analysis (shown in Figure.3), whereas Figure.11 evolves much the same as Figure.7

5.1. **Rate distortion performance.** The results of simulation for various rates are compared with that predicted by analysis. The results closely match.

The simulation performance without weights and with weights (weights are the scaling factors (see the problem description) used in the selection of picking a variable node. More precisely speaking, at every time step, a variable node is selected (from the available set of connected variable nodes from $G(t)$ with a probability, $\frac{w_i N_i(t)}{\sum_{j=0}^{3} w_j N_j(t)}$. With appropriate choice of weights, the distortion can be minimized. A good intuitive choice is to set $w_0 < w_1 < w_2 < w3$. This way,

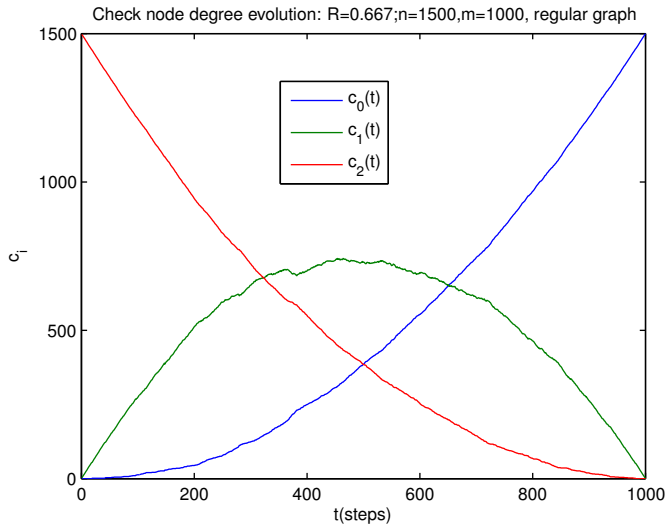

FIGURE 10. Simulation result: Result is matching with the analysis and result shown in Figure3. Evolution of the degree distribution of edges: The number of edges connected to check nodes of degree $i, i = 0, 1, 2$ is are shown, as it evolves with time. The x-axis is the time step. At start (i.e., $t = 0$), all the edges are connected to check nodes of degree 2. This refers to $c_0 = 0 = c_1 = 0, c_2 = n$. After m steps, all the edges are peeled off (edges are connected to none of the check nodes), which is equivalent to $c_0 = n$. Simulation carried out on regular graph with $n = 1500$,$R = 2/3$,$m = 1000$. The check node degree is 2 and variable node degree 3.

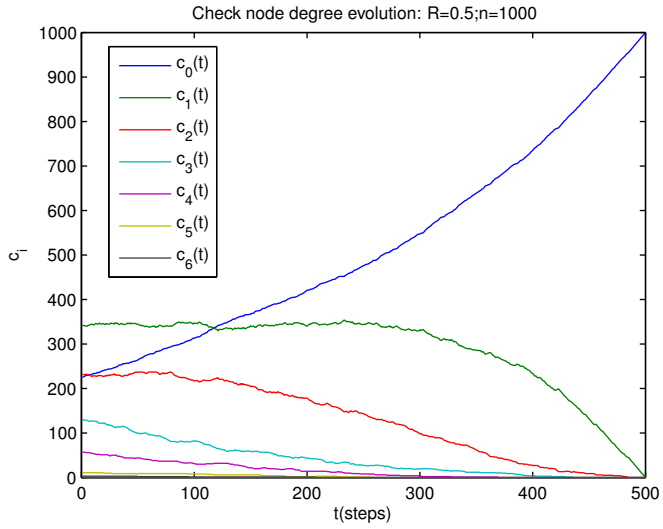

FIGURE 11. Simulation result: Check node degree evolution for the graph. The variable node degree is fixed but the check node degree follow a Poisson distribution. After m steps the algorithm stops. At that stage, all the edges are peeled off (edges are connected to none of the check nodes), which is equivalent to $c_0 = n$. Simulation carried out on such a graph with $n = 1000$,$R = 1/2$,$m = 500$. The check node degree follows a Poisson distribution and variable node degree equals 3.

the likelihood of variable node of higher type is improved (and thereby reducing the distortion). When simulated with $w_0 = 1, w_1 = 10, w_2 = 100, w_3 = 1000$, the distortion has come down considerably, predominantly at higher rates. The rational for choosing the weights in ascending order is to maximize the chance of picking a variable node of higher type, which helps to settle more number of variable node values and thus minimum Hamming distortion.

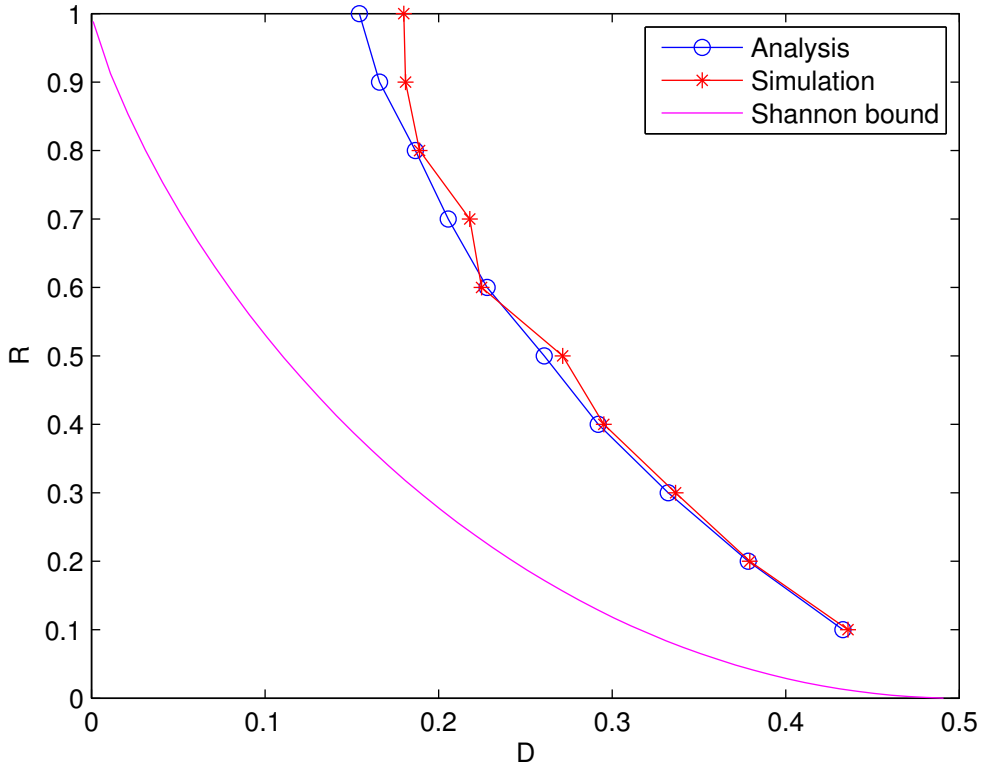

FIGURE 12. Performance analysis: Rate distortion curve for binary Bernoulli source. The analysis and simulation results provided. The result of analysis is closely match with the finite length simulation ($n = 1000$,$m = 500$, with 10 iterations) for an irregular graph (the initial input degrees form a Poisson distribution). The variable node degree is 3. The results provided here are with no weight (uniform weights $w_0 = w_1 = w_2 = w_3 = 1$. With larger n, a smoother curve is expected.

6. EXPERIMENTS AND POTENTIAL LEADS TO FUTURE WORK

Some of the things observed.

(1) When the check node distribution is fixed (not Poisson), the distortion performance appears to be slightly superior compared to Poissonian. This is what is observed with $(2, 3)$ regular graph with $n = 1000$. Again the improvement is negligible with low rates.
(2) enumerate
(3) What if the encoding does not follow Poisson distribution? Can we find a distribution which is optimal? Instead of following a uniformly at random node selection strategy, say we devise a more optimal strategy? Perhaps, we can find better performance?
(4) Instead of peeling and chopping of the nodes and edges at every step, if a message passing algorithm is used, would the performance improve?. Perhaps higher depth of iteration provide results close to the Shannon bound.In the current algorithm, there is certain amount of hard decision done at every stage, which result in higher distortion than optimum!
(5) Came across many papers providing bounds on performance on various kinds (Some schemes such as Survey propagation, maximum likelihood etc), but didn't have time to try out any of them.

7. CONCLUDING REMARKS

The simple noisy compression scheme using LDGM is analysed and simulated. Using the Wormald method, it is easy to analyze the scheme. The analysis result appears to be closely matching with the simulation for finite values of n. The rate distortion curve is plotted for both analysis and simulation. The performance of the algorithm improves with appropriate

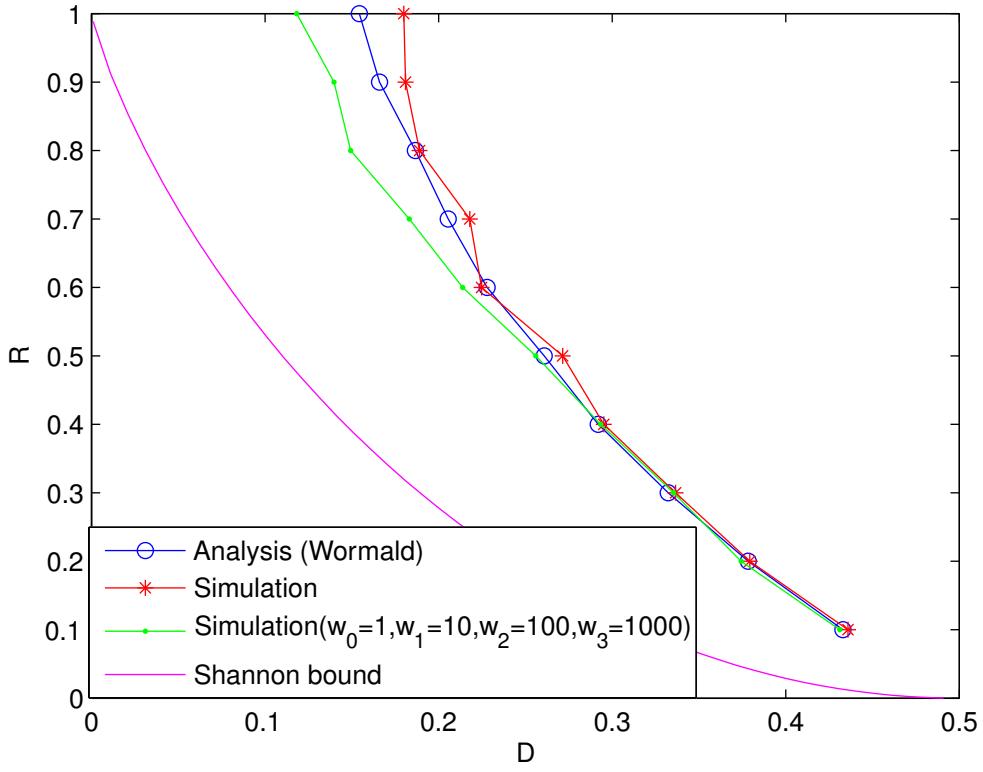

FIGURE 13. Performance analysis: Rate distortion curve for binary Bernoulli source ($p = 1/2$). The performance improvement with optimum weight selection is captured here. When weights $w_0 = 1, w_1 = 10, w_2 = 100, w_3 = 1000$ are chosen, the distortion reduces compared to that of uniform weights. The improvement is considerable at higher rates. The analysis results are provided only for the uniform case, but similar analysis can be done with weights as well. Simulations results, are based on limited number of iterations with finite values of n and m. $n = 100$ is used with 100 different realizations of the input distribution. With larger n, a smoother curve is expected.

weights on the variable node selection distribution. The performance of this simple algorithm is promising, even though it is not yet close to the ultimate rate distortion bound. However, with lower rates, the gap is significantly small.

8. Simulation Program

The simulation of the algorithm using matlab is provided here. The mathematica notebook is not inserted since the size is getting bigger.

```
% ---------------LDGM: Noisy data compression------
% This script simulate the algorithm [1] for nosy data compression using
% LDGM (Low density generator matrix) code [2]. Only the encoding is modelled
% here. The details of the result and algorithm please refer to the report
% [3].
% References:
% [1] Takehome Exam: Handout 10: Modern Coding Theory Course, 2008, EPFL
% [2] T.Richardson, R. Urbanke,Modern Coding Theory, Camridge Univ. Press 2007.
% [3] R.Pulikkoonattu, Report: Final Exam, Modern Coding theory, 2008
% Rethnakaran Pulikkoonattu
% This program isd developed as part of the final exam, Modern Coding
% theory, EPFL, 2008
% History: 2008/06/02 Created
%               2008/06/06 Verified
% Status: Working as expected (Not speed/memory optimized)

clear all;close all;

n=500;
M=n*[20:-1:1]/20;  % sets of m values to simulate
dv=3;                          % Degree of all variable nodes
d=[];                            % distortion
L=15;                          % Number of different input source to average
for ii=1:length(M)
    d=[];

    for l=1:L
        sword=randint(1,n);    % random binary source stream
        xword=sword;           % initialization!
%         x=-0*ones(1,n);
        m=M(ii);               % chosen m value for this iteration
        z=-0*ones(1,m);        % Initialization
        h1 = waitbar(0,'LDGM Simulation...');
        ve=[];
        %---------------Generate LDGM Matrix-----------------------
        for i=1:m
            cn=randperm(n);     % randomize indices of c.nodes connected to v.node m
            ve=[ve;cn(1:3)];    % The index of check nodes connected to variable nodes
        end

        G0=zeros(n,m);          % Generator matrix initialization. Place holder
        for cn=1:n
            [a,b]=find(ve==cn); % Find the v-node to which c-node cn is connected
            G0(cn,a)=1;         % The connected edge is marked as 1 in the LDGM matrix
        end

        G=G0;                   % Preserve the original generator matrix for later use
        %----------------LDGM matrix generated-------------------------

        tmax=m;                 % Number of steps for the algorithm/iteration to run
        tp=[];                  % initialize tp. helps when many m values are simulated
```

```
sh=[];                    % Debug only
for t=1:tmax              % Iterate for time steps
    waitbar(t/tmax,h1,'LDGM Simulation progressing...');
    tp=-1*ones(1,m);      % Initialize with all -1 vector
    for k=1:m             % For every v.node
        % ----------------- determine the type----------------------
        h=find(G(:,k)==1);% Find the connected check nodes: (h,cn) has ones
        w=G(h,:);         % identify the v.nodes connected to the check nodes
        sw=sum(w,2);      % the number of connected v.nodes for each of the
        %                   connected check nodes emanating from v.node k
        if ~isempty(sw)   % If there c.nodes connected (not isolated v.nodes)
            tp(k)=sum(sw==1);% type=num of c.nodes in residual graph of deg-1
        end
    end

    N0=sum(tp==0);
    N1=sum(tp==1);
    N2=sum(tp==2);
    N3=sum(tp==3);
    w0=1;
    w1=100;
    w2=100;
    w3=1000;
    ht=[zeros(1,round(N0*w0))...
        ones(1,round(N1*w1)) ...
        2*ones(1,round(N2*w2)) ...
        3*ones(1,round(N3*w3))]; % Make a distribution for the v.nodes
                                 % in the residual graph
    permht=randperm(length(ht)); % randomize the v.nodes
    bi=ht(permht(1));            % Select the first of permht, which is
                                 % essentially picking one random type
                                 % based on the type distribution ht
    v0=find(tp==0);              % List of Type 0 nodes, empty if none
    v1=find(tp==1);              % List of Type 1 nodes, empty if none
    v2=find(tp==2);              % List of Type 2 nodes, empty if none
    v3=find(tp==3);              % List of Type 3 nodes, empty if none

    switch bi                   % Based on the type of node,select a
                                % random v-node, from the corresponding
                                % available list
        case 0
            pp=randperm(length(v0));
            sv=v0(pp(1));
        case 1
            pp=randperm(length(v1));
            sv=v1(pp(1));
        case 2
            pp=randperm(length(v2));
            sv=v2(pp(1));
        case 3
            pp=randperm(length(v3));
            sv=v3(pp(1));
        otherwise
            disp('There appear to be a problem; Time to debug!');
```

```
            end
            clear ss;
            clear sc;
            clear ac;
            clear uc;

            sh=[sh sv]; % Debug
            ss=(find(G(:,sv)==1));  % list all check nodes connected to the
                                    % selected variable node
            sc=G(ss,:);             % Identify the variable nodes connected
                                    % to the check nodes
            ac=sum(sc,2);           % The total number of connected v.nodes
                                    % for each of the connected check nodes
                                    % emanating from v.node k
            uc_i=find(ac==1);
            uc=ss(uc_i);            % List of check nodes in the residual
                                    % graph which has degree-1

            if isempty(uc)          % If there are no c.nodes of degree 1 in
                                    % the residual graph,
                z(sv)=randint(1);   % If there is no majority, then assign random
                xword(ss)=mod(xword(ss)+z(sv),2); % Update c.nodes:Add (XOR)
                                                  % the value z(sv) to all (ALL)
                                                  % connected check nodes
                G(ss,sv)=0;         % Modify graph: Delete all the connected edges
            else
                sd=xword(uc);       % Get data at deg-1 v.nodes of resid-graph
                if (sum(sd)==length(sd)/2)
                    z(sv)=randint(1);           % If no majority assign randomly
                else
                    z(sv)=sum(sd)>length(sd)/2; % Update v.nodes: Assign the majority
                                                % value of connected v.nodes of degree
                                                % 1 in the residual graph to the
                                                % connected v.node
                end
                xword(ss)=mod(xword(ss)+z(sv),2); % Update c.nodes:Add (XOR) the value
                                                  % z(sv) to connected check nodes
                G(ss,sv)=0;                       % Modify G: Delete connected edges
            end

            [hc(t,:) hci(t,:)]=hist(sum(G,2),n);
            [hv(t,:) hvi(t,:)]=hist(sum(G,1),n);
        end
        close(h1);

        d=[d sum(xword)/n];% sum(xword)/n=sum(mod(sword'+mod(G0*z',2),2))/n;
        davg=mean(d)
        dmin=min(d);

    end

    D(ii)=davg;
    Dmin(ii)=dmin;
end
```

```
% Display
R=M/n

figure;
plot(D,R,'-bo');hold on;
y=0.001:0.01:0.5;
plot(y,1-(-y.*log2(y)-(1-y).*log2(1-y)),'m');
xlabel('D');
ylabel('R');
legend('Simulation','Shannon bound')

if (0)
    figure;
    R=M/n
    plot(d1,R,'-bo');hold on;
    y=0.001:0.01:0.5;
    plot(y,1-(-y.*log2(y)-(1-y).*log2(1-y)),'m');

    figure;
    hist(sum(G0,2),100);
    title('check node distribution (Poisson)');
end

%[mod(G0*z',2), xword' sword']
% G0*z'=sword+d(xword,sword); % Relationship. When d(s,x)=0, G0*z=s, ideal
% compression achieved here.
```

REFERENCES

[1] Modern Coding theory (R.Urbanke), EPFL Course (2008), Handout of Final Exam, 2008/06/02
[2] T.Richardson, R.Urbanke, Modern Coding theory, Cambridge University Press, 2007.
[3] T.Cover, J.Thomas, Elements of Information theory, 2nd Edition, Wiley International, 2006
[4] N.C. Wormald, Differential equations for random processes and random graphs, Annals of Applied Probability 5 (1995), 1217-1235.
[5] N.C. Wormald, The differential equation method for random graph processes and greedy algorithms, in Lectures on Approximation and Randomized Algorithms (M. Karonski and H.J. Proemel, eds), pp. 73-155. PWN, Warsaw, 1999.
[6] W. Duckworth and N.C. Wormald, On the independent domination number of random regular graphs, Combinatorics, Probability and Computing 15 (2006), 513-522.
[7] E. Martinian and J. Yedidia, "Iterative quantization using codes on graphs", in Proc. of the Allerton Conf. on Commun., Control, and Computing, Oct. 2003.
[8] A. Dimakis, M. Wainwright, and K. Ramchandran, "Lower bounds on the rate-distortion function of LDGM codes", in Proc. of the IEEE Inform.Theory Workshop, 2007.
[9] M. J. Wainwright and E. Maneva, "Lossy source coding via message passing and decimation over generalized codewords of LDGM codes",in Proc. of the IEEE Int. Symposium on Inform. Theory, Adelaide, Australia, Sept. 2005, pp. 1493-1497.
[10] E. Martinian and M. J. Wainwright, Low-density codes achieve the rate-distortion bound, in Proc. of the Data Compression Conference,Snowbird, UT, Mar. 2006.
[11] R. Pulikkoonattu, Homework assignment-8 Preferential attachment, Modern Coding theory (R.Urbanke), EPFL Course, EPFL 2008.
[12] R.Pulikkoonattu, Final exam (Modern Coding theory course, EPFL 2008) Mathematica notebook

E-mail address: rethnakaran.pulikkoonattu@epfl.ch
E-mail address: ratnuu@gmail.com