A CLASS OF STRUCTURED LDPC CODES WITH LARGE GIRTH

Jin Lu, José M. F. Moura, and Urs Niesen

Department of Electrical and Computer Engineering
Carnegie Mellon University, Pittsburgh, PA 15213
jinlu, moura@ece.cmu.edu

ABSTRACT

We introduce a class of structured LDPC codes—turbo-structured LDPC (TS-LDPC) codes—composed of two subtrees connected by an interleaver. TS-LDPC codes with good girth properties are easy to design: careful design of the interleaver component prevents short cycles in its Tanner graph. We present a methodology to design TS-LDPC codes with arbitrary column weight $j \geq 2$ and arbitrary girth. In addition, we describe a complexity reduced decoding algorithm. Simulation results demonstrate the good performance of TS-LDPC codes when compared to random LDPC codes of the similar size and rate.

1. INTRODUCTION

Low-density parity-check (LDPC) codes [1], can be applied in numerous tasks, e.g., communication systems, magnetic recording channels. Their performance is close to the Shannon limit using iterative decoding [2].

An LDPC code can be described by a bipartite graph called Tanner graph [3]. The length of the shortest cycle in a Tanner graph is referred to as its girth g. Short cycles in Tanner graphs tax the computing effort of the decoding algorithm and prevent it from converging to the optimal decoding result. Furthermore, reference [3] derives a lower bound on the minimum distance d_{min}. This lower bound increases exponentially with girth. Therefore, LDPC codes with good girth properties are particularly desirable.

Recently, cyclic and quasi-cyclic LDPC codes have drawn much attention in the sense that they facilitate low-complexity encoder and decoder designs. However, they have limited girths. Tanner, [4], proved that the girth of such codes with column weight $j \geq 3$ is less than or equal to 6. This prevents the girth of (n, j, k) cyclic and quasi-cyclic LDPC codes of growing as $\frac{\log n}{\log[(j-1)(k-1)]}$, [1], hence performing poorly at very long code block length.

We present a new type of structured LDPC codes with arbitrary girth that are stimulated from turbo designs, which we refer to as *turbo-structured* LDPC codes (TS-LDPC). The Tanner graph of such codes is composed of two subtrees that are connected in a turbo-like manner by an interleaver. This structure facilitates the systematic design of LDPC codes with large girth and flexible code rates. Our turbo-structured LDPC codes are distinct from the codes in [5] that are also inspired by turbo codes in their design of concatenated tree codes. However, our TS-LDPC codes are turbo for the decoder side, while [5] is turbo for the encoder side. The resulting TS-LDPC codes and the concatenated tree codes are very different, see [6] for comments.

This work was supported by the Data Storage Systems Center (DSSC) at Carnegie Mellon University, and by NSF grant # ECS-0225449.

2. TS-LDPC CODES

The Tanner graph of TS-LDPC codes is composed of three components: two height-balanced trees, denoted as an upper-tree T_U, and a lower-tree T_L, and an interlaver that connect T_U and T_L. The leaf-nodes of T_U are bit nodes whereas the leaf-nodes of T_L are check nodes. Both trees T_U and T_L contain h tiers (or layers). The first tier of T_U contains only one check node C^*—the root, as shown in Figure 1. To match T_U, we let the root of T_L to be a bit node V^* and connect V^* to C^*. The two trees are "coupled" in a turbo-like manner such that many edges join the leaf-nodes of T_U and T_L together, see Figure 1. The structure formed by the edges connecting the leaf-nodes of T_U and the leaf nodes of T_L is named the interleaver $\mathcal{I}$. Since we are interested in regular LDPC codes, we let all the bit nodes have uniform degree j and all the check nodes have uniform degree k. For example, a TS-LDPC code with $h = 4$, $j = 3$, and $k = 4$ is depicted in Figure 1. Ignoring possible dependency of a few rows in the parity-check matrix, the code rate is $\rho = 1 - \frac{j}{k}$. We can easily choose the values of j and k to create a TS-LDPC code with the desired code rate.

3. INTERLEAVER DESIGN

By construction, each leaf-node in T_U is connected to $q = j - 1$ leaf-nodes in T_L. This is a one-to-q mapping, while the standard interleaver is a one-to-one mapping between elements of two sets with the same size. To get a standard

interleaver, we introduce "auxiliary nodes" (solid triangles) as shown in Figure 2 to facilitate the code design. For each leaf-node in T_U, we add $j-1$ auxiliary nodes as its children. Similarly, each leaf-node in T_L has $k-1$ auxiliary nodes as its descendants.

After introducing "auxiliary nodes," we notice that cycles present in the codes must contain at least four "auxiliary nodes"—two auxiliary nodes of T_U and two auxiliary nodes of T_L. We classify cycles into two disjoint categories: Type-I cycles contain four and only four auxiliary nodes; type-II cycles contain more than four auxiliary nodes. We will dispose of them separately.

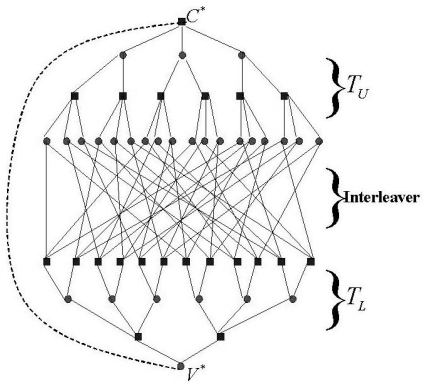

Fig. 1. TS-LDPC code: Tanner graph, $h = 4$, $j = 3$, $k = 4$.

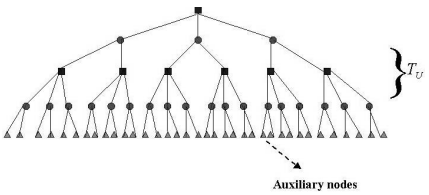

Fig. 2. Auxiliary nodes in the upper tree T_U

For T_U with h tiers, there are $[(k-1)(j-1)]^{h/2}$ auxiliary nodes in T_U. We address the interleaver design problem algebraically by indexing all the auxiliary nodes of T_U, from 0 to $[(k-1)(j-1)]^{h/2}-1$ in a format we explain next that we refer to as *p-q-alternate-decimal*, where $p = k-1$ and $q = j-1$. We need h digits in the p-q-alternate-decimal indexing and number these h digits from 1 to h, starting from the rightmost one. The odd digits take values 0 to $q-1$ and the even digits take values 0 to $p-1$. Similarly, we index all the auxiliary nodes of T_L from 0 to $[(k-1)(j-1)]^{h/2}-1$ and represent all these indices in *q-p-alternate-decimal* format. We provide an example. With reference to the index in Figure 2, its index X_{p-q} in p-q-alternate-decimal form is:

$$X_{p-q} = \overbrace{x_4}^{p}\overbrace{x_3}^{q}\overbrace{x_2}^{p}\overbrace{x_1}^{q} \quad (1)$$

In equation (1), x_i, $i = 1 \cdots 4$, represents the i^{th} digit. The corresponding value of X_{p-q} in decimals is

$$X_{p-q} = (x_4 \times pq^2 + x_3 \times pq + x_2 \times q + x_1)_{10}.$$

3.1. Type-I Cycles: Digit-wise Reversal

We consider first how to avoid short type-I cycles. We start with a simple interleaver design—*digit-wise reversal*. For an index X_{p-q} in p-q-alternate-decimal form with h digits, its digit-wise reversal interchanges the i^{th} digit and the $(h+1-i)^{\text{th}}$ digit. We represent the digit-wise reversal operator by $\pi_d(\cdot)$. For the index X_{p-q} in equation (1), its digit-wise reversal is:

$$\begin{aligned}\pi_d(X_{p-q}) &= \overbrace{x_1}^{q}\overbrace{x_2}^{p}\overbrace{x_3}^{q}\overbrace{x_4}^{p} \\ &= (x_1 \times qp^2 + x_2 \times qp + x_3 \times p + x_4)_{10}\end{aligned}$$

We state the advantage of the digit-wise reversal interleaver in the following theorem and omit the proof. For a detailed proof, please refer to [7].

Theorem 1 *Connecting the auxiliary nodes indexed by X_{p-q} in T_U to the auxiliary nodes indexed by $\pi_d(X_{p-q})$ in T_L guarantees the resulting type-I cycles is at least of length $2h$ (h denotes the number of tiers in T_U).*

Following theorem 1, we can enlarge the length of type-I cycles by simply increasing tiers of sub-trees.

3.2. Type-II 4-Cycles

Theorem 1 prevents short type-I cycles, however, it may lead to many short type-II cycles, even type-II 4-cycles.

To avoid short type-II cycles, we propose a method called *grouping and shifting*. The *shift* S is defined to be a constant in q-p-alternate-decimal format that is added to the original index $\pi_d(X_{p-q})$ to form a new index. For example, let $\pi_d(X_{p-q}) = \overline{x_1x_2x_3x_4}$, shift $S = \overline{s_1s_2s_3s_4}$, and $\dot{+}$ represent the digit-wise addition, then

$$\pi_d(X_{p-q}) \dot{+} S = \overline{y_1y_2y_3y_4} \quad (2)$$

In equation (2), $y_i = x_i \dot{+} s_i = \bmod(x_i + s_i, \text{div}_i)$. where $\text{div}_i = p$ if i is even and $\text{div}_i = q$ if i is odd. Similarly, we represent the digit-wise subtraction by $\dot{-}$.

Next, we divide all the auxiliary nodes of T_U into $k-1$ groups, grouping together those nodes with the same leftmost digit in their p-q-alternate-decimated indices. The auxiliary nodes of T_L can, likewise, be classified into $j-1$ different groups based also on the values of the leftmost digits

in their q-p-alternate-decimated indices. We further let the shift to be the same when we connect the auxiliary nodes of T_U in the same group to the auxiliary nodes of T_L in the same group. Denote by $S_{y,z}$ the shift introduced when we connect the auxiliary nodes of T_L in the y^{th} group to the auxiliary nodes of T_U in the z^{th} group. For different y and z, the shifts $S_{y,z}$ may be the same or different from each other. In addition, since $S_{y,z}$ is not to affect the assigning of the groups, its leftmost and rightmost digits are always set to 0.

The following theorem proved elsewhere relates type-I cycles to shifts:

Theorem 2 (TYPE-I 2h-CYCLES) *Connecting the auxiliary node indexed by X_{p-q} in the z^{th} group in T_U to the auxiliary node indexed by $\pi_d(X_{p-q})\dot{+}S_{y,z}$ ($S_{y,z}$ can be any possible value) in the y^{th} group in T_L guarantees that any type-I cycle formed is at least of length $2h$ (h denotes the number of tiers in T_U).*

Hence, when following theorem 2, we have $(j-1)(k-1)$ free parameters $S_{y,z}$ to adjust to prevent short type-II cycles while in the mean time all short type-I cycles are avoided.

Theorem 3 provides a sufficient condition on the shifts $S_{y,z}$ that can be used to prevent type-II 4-cycles.

Theorem 3 (NO TYPE-II 4-CYCLES) *Consider the m^{th} and the n^{th} groups of nodes in the upper-tree T_U and the i^{th} and the j^{th} groups in the lower-tree T_L. If the associated shifts satisfy $S_{i,m}\dot{+}S_{j,n} \neq S_{i,n}\dot{+}S_{j,m}$, then NO type-II 4-cycles are formed between these four groups.*

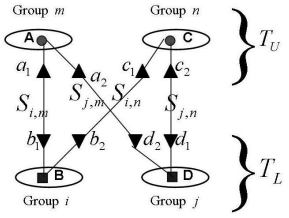

Fig. 3. (a)

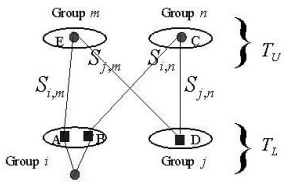

Fig. 3. (b)

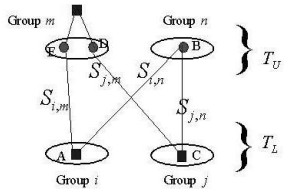

Fig. 3. (c)

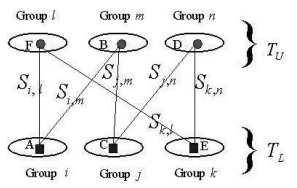

Fig. 3. (d)

Fig. 3.

Proof: We prove theorem 3 by proving an equivalent proposition: If a type-II 4-cycle is formed, then the associated shifts MUST satisfy $S_{i,m}\dot{+}S_{j,n} = S_{i,n}\dot{+}S_{j,m}$. Figure 3.(a) shows a type-II 4-cycle containing four vertices A, B, C, and D. It can also be viewed as a cycle containing eight auxiliary nodes a_1, a_2, c_1, c_2, b_1, b_2, d_2, and d_1. With reference to the plot in Figure 3.(a), and assuming that the index for the auxiliary node a_1 is X_{a_1}, then according to the connecting rule presented in theorem 2, the index for the auxiliary node b_1 is $X_{b_1} = \pi_d(X_{a_1})\dot{+}S_{i,m}$. Let δ_b denote the difference between X_{b_1} and X_{b_2}, i.e., $\delta_b = X_{b_2}\dot{-}X_{b_1}$. Since the auxiliary nodes b_1 and b_2 are connected to the same check node B, only the rightmost digit of the q-p-alternate-decimal form of δ_b is non-zero, all its other digits are zero. The index for the auxiliary node c_1 is $X_{c_1} = \pi_d(X_{b_2}\dot{-}S_{i,n})$. Again, let $\delta_c = X_{c_2} - X_{c_1}$. Then the $p-q$-alternate-decimal form of δ_c has only one non-zero digit, its rightmost digit. The index for the auxiliary node d_1 is $X_{d_1} = \pi_d(X_{c_2})\dot{+}S_{j,n}$. Let $X_{d_2} = X_{d_1} + \delta_d$; the index for the auxiliary node a_2 in terms of X_{d_2} is $X_{a_2} = \pi_d(X_{d_2}\dot{-}S_{j,m})$. The relationship between X_{a_1} and X_{a_2} is $X_{a_1} = X_{a_2} + \delta_a$. Iterating in the definition of X_{a_2}, we have

$$\begin{aligned}\pi_d[\pi_d(\pi_d(\pi_d(X_{a_1})\dot{+}S_{i,m}\dot{+}\delta_b\dot{-}S_{i,n})\dot{+}\delta_c)\\ \dot{+}S_{j,n}\dot{+}\delta_d\dot{-}S_{j,m}] + \delta_a = X_{a_1}.\end{aligned} \quad (3)$$

As $\pi_d(\pi_d(X)) = X$, this leads to

$$\pi_d(S_{i,m}\dot{-}S_{i,n}\dot{+}S_{j,n}\dot{-}S_{j,m})\dot{+}[\pi_d(\delta_b\dot{+}\delta_d)\dot{+}\delta_c\dot{+}\delta_a] = 0 \quad (4)$$

In the q-p-alternate-decimal form of a shift, the leftmost and the rightmost digits are always zero. On the other hand, except for the leftmost and rightmost digits, all the digits in δ_c, δ_a, $\pi_d(\delta_b)$, and $\pi_d(\delta_d)$ are zero. Therefore, equation (4) can be split into two sub-equations:

$$\pi_d(S_{i,m}\dot{-}S_{i,n}\dot{+}S_{j,n}\dot{-}S_{j,m}) = 0 \quad (5)$$

$$\pi_d(\delta_b\dot{+}\delta_d)\dot{+}\delta_c\dot{+}\delta_a = 0. \quad (6)$$

It follows then from equation (5)

$$S_{i,m}\dot{+}S_{j,n} = S_{i,n}\dot{+}S_{j,m}. \quad (7)$$

This completes the proof.

3.3. Exclude Type-II Cycles of Arbitrary Length

We observe that there are many different classes of type-II cycles. For example, all type-II cycles of length six can be divided into three classes, as shown in Figure 3.(b), 3.(c), and 3.(d). For convenience, we divide all type-II cycles up to length $2L$ into $L-1$ equivalence sets based on the number of edges N_I contained in the interleaver component. For example, all type-II cycles of length six belong to two equivalence sets: Cycles shown in Figure 3.(b) and 3.(c), with four edges in the interleaver, are in the same equivalence set. Another equivalence set contains the cycles shown in

Figure 3.(d) that have six edges in the interleaver. The following theorem eliminates the cycles in each equivalence set.

Each cycle in the equivalence set with $N_I = 2k$ is associated with $2k$ shifts $S_{y_1,z_1}\ S_{y_2,z_2}\ \ldots\ S_{y_{2k},z_{2k}}$ whose subindices satisfy the following two conditions:

(i) $y_{2t-1} = y_{2t}$, $t = 1, 2, 3, \ldots, k$ and $z_{2t} = z_{2t+1}$, $t = 1, 2, 3, \ldots, k-1$

(ii) $y_{2t} \neq y_{2t+1}$, $t = 1, 2, 3, \ldots, k-1$ and $z_{2t-1} \neq z_{2t}$, $t = 1, 2, 3, \ldots, k$

Define the shift matrix $\mathbf{S} = [S_{y,z}]$ to be a matrix that collects all the shifts. From the point of view of graph theory, $S_{y_1,z_1}, S_{y_2,z_2}, \ldots, S_{y_{2k},z_{2k}}$ are adjacent vertices of a closed path in the $\mathbf{S}$ matrix. For example, as shown in Figure 4, $S_{1,1}$, $S_{1,2}$, $S_{2,2}$, and $S_{2,1}$ are vertices of a closed path of length four (dashed line) in $\mathbf{S}$ and $S_{2,5}$, $S_{2,6}$, $S_{4,6}$, $S_{4,4}$, $S_{3,4}$, and $S_{3,5}$ are vertices of a closed path of length six (solid line) in $\mathbf{S}$.

Theorem 4 *For the equivalence set with $N_I = 2k$, let $\Delta S = \sum_{u=1}^{k} S_{y_{2u},z_{2u}} - \sum_{v=1}^{k} S_{y_{2v-1},z_{2v-1}}$. Each ΔS having h digits is free of any cycle having less than $N_T = 2l$ edges in the trees (both T_U and T_L) if it contains at most $d = h-l-2$ consecutive digits "0" in its q-p-alternate-decimal expansion.*

The proof is rather lengthy. We prove it elsewhere.

By choosing suitable shifts $S_{y,z}$ according to theorems 2-4, we can avoid all short type-I and type-II cycles up to the desired length $g - 2$.

$$\mathbf{S} = \begin{bmatrix} S_{1,1} & S_{1,2} & S_{1,3} & S_{1,4} & S_{1,5} & S_{1,6} \\ S_{2,1} & S_{2,2} & S_{2,3} & S_{2,4} & S_{2,5} & S_{2,6} \\ S_{3,1} & S_{3,2} & S_{3,3} & S_{3,4} & S_{3,5} & S_{3,6} \\ S_{4,1} & S_{4,2} & S_{4,3} & S_{4,4} & S_{4,5} & S_{4,6} \end{bmatrix}$$

Fig. 4. Two closed paths in the shift matrix $\mathbf{S}$

As an illustration, we applied the above methods to construct a (6666, 3, 6) regular LDPC code, rate .5, with girth $g = 10$. Its structure is given by the 3333×6666 matrix $\mathbf{H}$ shown in Figure 5. We can clearly identify T_U, T_L, and the interleaver component $\mathcal{I}$ from the constructed matrix, as labelled in Figure 5. In this matrix, along the solid lines, there is a single 1 in each row, while along the dashed thicker diagonals there are five 1's in each row, so that per row there are six 1's.

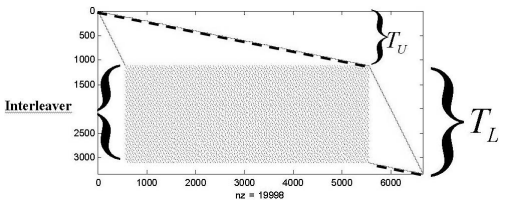

Fig. 5. Parity-check matrix of a (6666, 3, 6) TS-LDPC code with girth 10

4. PERFORMANCE EVALUATION

We compare by simulation the bit error rate (BER) of the TS-LDPC codes with the BER of randomly constructed LDPC codes that are free of 4-cycles [2] in additive white Gauss noise (AWGN) channels. The codes are decoded with the sum-product algorithm [9], and we adopt the signal to noise ratio (SNR) defined in [2]: SNR $= 10\log_{10}\left[E_b/\left(2r\sigma^2\right)\right]$ where r denotes the code rate.

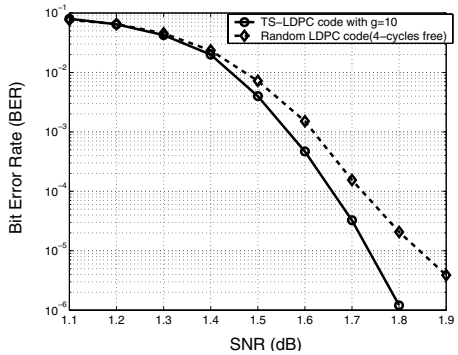

Fig. 6. BER performance comparison between a column weight $j = 3$ TS-LDPC code with girth 10 and a randomly constructed LDPC code of the same size free of 4-cycles.

The plot in Figure 6 shows the BER performance for a column weight $j = 3$ TS-LDPC code with girth 10 (solid line). For comparison, we also show the BER performance of a randomly constructed LDPC code (no 4-cycles) with column weight $j = 3$ (dashed line). Both codes have the same block length 6666 and the same code rate 1/2.

From Figure 6, it can be seen that the BER performance of the TS-LDPC code is 0.12dB better than that of the random LDPC code at BER$= 10^{-5}$ while at low SNR both codes have similar error-correcting performance. According to [3], the $(6666, 3, 6)$ TS-LDPC code has minimum distance $d_{min} \geq 10$. Since the lower bound of d_{min} derived in [3] is not tight, the actual d_{min} of the $(6666, 3, 6)$ TS-LDPC code may be much larger than 10. In the high SNR region, $d_{\min}$ is a dominant factor in determining the code BER performance. This explains why the TS-LDPC

code with girth 10 has good BER performance in the high SNR region.

5. FAST DECODING

The TS-LDPC codes can be effectively decoded by what we refer to as the *turbo like decoding algorithm (TLDA)* [8]. The TLDA is as follows:

Step 1: Initialization.

Step 2: Decode T_U, using the updated information provided by $\mathcal{I}$, and, after decoding, transmit the new probabilistic information to T_L through $\mathcal{I}$.

Step 3: Decode T_L, using the updated information from $\mathcal{I}$ and then transmit the new probabilistic information back to T_U through $\mathcal{I}$.

Step 4: Compute the temporary decoding outputs. If decoding success is achieved, go to step 5. Otherwise, go back to step 2.

Step 5: End.

The message updatings of the TLDA are the same as those for the sum-product decoding algorithm [1].

We comment briefly on the advantages of TLDA. It is well known, [9], that with a cycle-free Tanner graph, the sum-product algorithm terminates in a finite number of steps and yields minimum symbol error probability. Therefore, in isolation, the local decoding for each cycle-free component is optimal. The TLDA is still iterative: each component transmits its *a posteriori probability* (APP) information to the others through the interleaver and, in turn, these components use these APPs as a priori information to start their own decoding process.

We present a simulation study by 20000 Monte Carlo simulations comparing the performance of the code using TLDA and the standard sum-product decoding. This is a $(6666, 3, 6)$ TS-LDPC code with uniform column weight $j = 3$ and code rate 0.5. Figure 7.(a) shows the BER performance of the code. The plots in Figure 7.(b) show that TLDA converges faster by about 50% (smaller number of iteration steps) than the sum-product decoding algorithm.

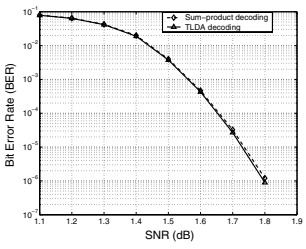

Fig. 7. (a)

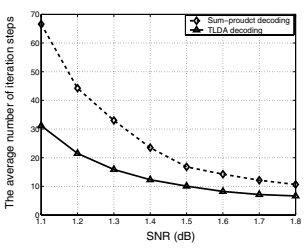

Fig. 7. (b)

Fig. 7.

6. CONCLUSION

In this paper, we propose a new class of well-structured LDPC codes—TS-LDPC codes. We presented designs of flexible code rate with arbitray column weight and arbitrary girth. TS-LDPC codes can be decoded efficiently, characteristics that make them attractive in applications, e.g., communication systems and data storage systems.

7. REFERENCES

[1] R. G. Gallager, *Low-Density Parity Check Codes*, MIT Press, Cambridge, MA, 1963.

[2] D. J. C. Mackay", "Good error-correcting codes based on very sparse matrices," *IEEE Trans. on Information Theory*, vol. 45, no. 2, March 1999, pp. 399–431.

[3] R. Michael Tanner, "A recursive approach to low complexity codes," *IEEE Trans. on Information Theory*, vol. IT-27, no. 5, Sep. 1981, pp. 533-547.

[4] R. Michael Tanner, "Spectral graphs for quasi-cyclic LDPC codes," in *Proceedings of International Symposium on Information Theory*, Washington, DC, June 24-29 2001.

[5] Li Ping and Keying Y. Wu, "Concatenated tree codes: A low-complexity, high-performance approac", in *IEEE Trans. Inform. Theory*, vol. 47, no.2, Feb. 2001, pp. 791–799.

[6] Jos M. F. Moura, J. Lu, H. Zhang, "Structured LDPC codes with large girth,", in *IEEE singal processing magazine*, vol. 21, no.1, Jan. 2004, pp. 42-55.

[7] J. Lu and José M. F. Moura, "Turbo design for LDPC codes with large girth," in *IEEE Signal Processing and Wireless Communications (SPAWC) Workshop*, Rome, Italy, June 2003.

[8] Jin Lu and José M. F. Moura, "Turbo Like Decoding of LDPC Codes", in *IEEE INTERMAG 2003*, Apr. 2003, Boston, MA.

[9] F. R. Kschischang, B. J. Frey, H. A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Transactions on Information Theory*, vol. 47, no.2, pp.498-519, Feb. 2001.