2003 4th IEEE Workshop on Signal Processing
Advances in Wireless Communications

TURBO DESIGN FOR LDPC CODES WITH LARGE GIRTH

Jin Lu and José M. F. Moura

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

ABSTRACT

LDPC codes, with performance extremely close to the theoretical limit, are gaining increased attention of the communication systems designers. LDPC codes with good girth properties are of particular interest. In this paper, we design large girth regular LDPC codes in a turbo-like manner. Specifically, we describe codes that are two sub-trees interconnected by an interleaver. Careful design of the interleaver block eliminates short cycles in its factor graph. We present designs for turbo-like LDPC codes with column weight $j \geq 3$ and girth at least 8 and with column weight $j = 2$ and girth at least 16. These proposed codes support a wide range of code rates and code block length, suitable for most practical applications.

1. INTRODUCTION

Low-density parity-check (LDPC) codes [1], error correcting codes based on very sparse parity check matrices, exhibit performance close to the Shannon limit using iterative decoding [2]. Extensive research on turbo codes and LDPC codes has shown that, when decoded by the iterative message passing algorithm, LDPC codes show better decoding performance than turbo codes as the block length increases. Moreover, the high computational complexity associated with the BCJR algorithm is a major drawback for using turbo codes, while LDPC codes can be iteratively decoded using the sum-product algorithm with comparatively less complexity. Therefore, LDPC codes are actively being considered in numerous applications, including magnetic recording channels, optical fiber transmission, or wireless communications (fixed or mobile). For example, [3] studies LDPC codes for communication systems with multiple antennas, and [4] applies LDPC codes to orthogonal frequency division multiplexing (OFDM) systems in different fading environments.

LDPC codes can be represented by a bipartite graph, its factor graph [5]. These are composed of two sets of nodes, namely, variable nodes and check nodes. Each variable node represents a bit in a code word and each check node corresponds to a parity-check constraint. If a variable node is constrained by a check node, there is an edge connecting these two nodes. The girth of the code, defined as the length of the shortest cycle in its factor graph, is a crucial parameter since large girth leads to reduced dependence in the message passing and more efficient iterative decoding when using the sum-product algorithm [6]. Moreover, large girth guarantees large minimum distance d_{min} between codewords, [7], therefore mitigating the error floor at high E_b/N_0. Hence, it is of increasing interest to design LDPC codes with good girth properties, [6].

In general, LDPC codes are generated by randomly constructing a low-density parity check matrix from a suitable ensemble, [8]. However, structured regular LDPC codes are particularly desirable to simplify the hardware implementation of the encoding and decoding systems.

Cyclic and quasi-cyclic LDPC codes [9], a type of structured LDPC codes, are of much recent interest, [6, 10]. They have low-complexity encoding and can be constructed algebraically. Though having relatively simple algebraic descriptions, their girth is quite limited. Tanner, [10], proved that all codes with column weight $j \geq 3$ whose parity check matrices are constructed from circulants must contain a 6-cycle, so the girth of such codes is at most 6. For column weight $j = 2$ quasi-cyclic LDPC codes, Tanner, [6], also showed that the girth of the corresponding factor graph cannot be greater than twelve. This implies that the codes will fail to show the logarithmic relationship between girth and the code block length, [1]. For sufficiently long lengths, random LDPC codes outperform cyclic and quasi-cyclic LDPC codes. Hence, their short girth properties seriously limits the application of cyclic or quasi-cyclic LDPC codes when very long codes are desired. Another drawback of such codes is that there are only a limited range of code lengths and code rates available, not flexible enough to be suitable in many tasks.

To overcome the above difficulties, we consider a new class of structured LDPC codes using turbo designs. The factor graphs of such codes contain two sub-trees that con-

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

nect to each other through an interleaver. This structure helps to create LDPC codes with large girth and flexible code rates. Another significant feature of turbo-like LDPC codes is that they require surprisingly small memory in its encoder and decoder design, which leads to low-complexity hardware implementations.

2. TURBO DESIGN OF LDPC CODES

Our designs are similar to the turbo structure: we interconnect two tree structures to create LDPC codes with large girth. The factor graph of such a code contains two height-balanced sub-trees, denoted as an upper-tree T_U, for which the leaf nodes are variable nodes, and a lower-tree T_L, for which the leaf nodes are check nodes. We represent the height of T_U and T_L, say, its number of tiers, by h. The first tier of T_U contains only one check node—the root, as shown in figure 1. On the other hand, the root of T_L (shown in figure 2) is a variable node. The two trees are "combined" in a turbo-like manner such that the leaf-nodes of T_L are connected to the leaf-nodes of T_U, see figure 3. We call the structure formed by the connecting edges between the leaf-nodes of T_L and T_U an "interleaver." All the variable nodes have uniform degree j and all the check nodes have the same degree k, except for the roots of T_U and T_L, whose degrees are set to be $k-1$ and $j-1$ respectively. For example, a turbo-like LDPC code with $h = 4$, $j = 3$ and $k = 4$ is shown in figure 3. To make the code exactly regular, we connect the root (a check node) of T_U and the root (a variable node) of T_L directly by an edge, hence forming a regular LDPC code shown in figure 3. It is easy to derive that the code rate $\rho = 1 - \frac{j}{k}$. For a given code rate ρ^*, just choose the two parameters j and k to satisfy the equation $\rho^* = 1 - \frac{j}{k}$: for example, for $\rho^* = 8/9$ and column weight 3, simply let $j = 3$ and $k = 27$ in each of the component trees.

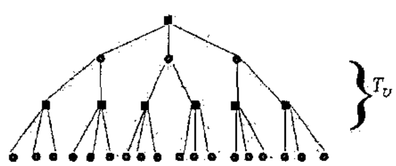

Fig. 1. Upper tree T_U of a turbo-like LDPC code with column weight 3, row weight 4 and height 4

The significance of the above structures lies in that it makes the design of large girth LDPC codes easy. As in isolation, no cycle exists in either of the sub-trees, the cycles in turbo-like LDPC codes are introduced by the interleaver. This paper discusses methods to devise interleavers that guarantee that the resulting turbo-like LDPC codes have

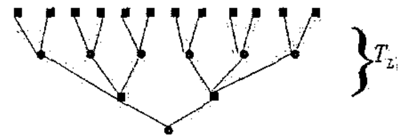

Fig. 2. Lower tree T_L of a turbo-like LDPC code with column weight 3, row weight 4, and height 4

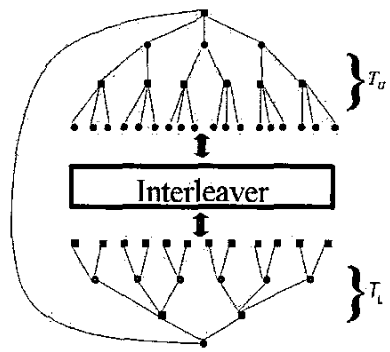

Fig. 3. Turbo-like LDPC code with column weight 3, row weight 4 and height 4

girth greater than or equal to 8. In particular, column weight 2 turbo-like LDPC codes can be further made free of any cycles with length less than 16.

3. CONSTRUCTING LARGE GIRTH LDPC CODES USING TURBO DESIGNS

Our goal in this section is to build turbo-like LDPC codes with large girth. For convenience, we introduce "auxiliary nodes" (represented by solid triangles) as shown in figure 4. For each leaf node in the upper tree T_U, since it connects to $j-1$ check nodes in the lower tree T_L, we add $j-1$ auxiliary nodes to it and let these auxiliary nodes be its descendants. Similarly, we introduce auxiliary nodes to the lower tree T_L such that each leaf node of T_L has $k-1$ auxiliary

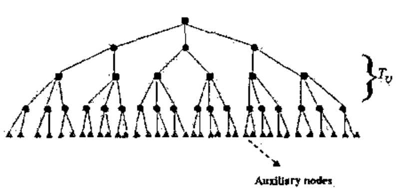

Fig. 4. Auxiliary nodes in a turbo-like LDPC code with column weight 3, row weight 4 and height 4

nodes as its descendants. There is a one to one correspondence between auxiliary nodes of T_U and auxiliary nodes of T_L. Figure 5 shows a path connecting T_U's leaf node

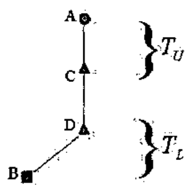

Fig. 5. Auxiliary nodes of T_U and T_L

A to T_L's leaf node B through T_U's auxiliary node C and T_L's auxiliary node D. That means, in the original factor graph, nodes A and B are directly connected by an edge. Therefore, our task can be equivalently expressed as finding an appropriate one-to-one mapping between auxiliary nodes of T_U and auxiliary nodes of T_L that guarantees large girth. As mentioned, no cycles exist in any of the sub-trees in isolation, so cycles present in the codes must contain "auxiliary nodes," which is at least four (two auxiliary nodes of T_U and two auxiliary nodes of T_L). We classify the cycles into two disjoint categories: type-I and type-II cycles, depending on how many auxiliary nodes a cycle contains. Type-I cycles contain four and only four auxiliary nodes and are denoted by C_I; type-II cycles contain more than four auxiliary nodes and are denoted by C_{II}.

Recall that k denotes the degree of the check nodes in a factor graph, or equivalently, the row weight of the corresponding parity-check matrix; j denotes the degree of the variable nodes. We introduce a *p-q-alternate decimal* to index all the auxiliary nodes of T_U and a *q-p-alternate decimal* to index all the auxiliary nodes of T_L, where $p = k-1$ and $q = j-1$. This will be useful in preventing short cycles. We introduce it through an illustration. Assume the p-q-alternate decimal representation of an index is $\overline{\underbrace{a_1}_{p}\underbrace{a_2}_{q}\underbrace{a_3}_{p}\underbrace{a_4}_{q}}$. It relates to its value in the decimal system as follows:

$$\overline{\underbrace{a_1}_{p}\underbrace{a_2}_{q}\underbrace{a_3}_{p}\underbrace{a_4}_{q}} = (a_1 \times pq^2 + a_2 \times pq + a_3 \times q + a_4)_{10}$$

where $0 \leq a_1 \leq p-1$, $0 \leq a_2 \leq q-1$, $0 \leq a_3 \leq p-1$, and $0 \leq a_4 \leq q-1$.

The *symbol-wise reversal* $\pi_S(\overline{\underbrace{a_1}_{p}\underbrace{a_2}_{q}\underbrace{a_3}_{p}\underbrace{a_4}_{q}})$ is defined as $\pi_S(\overline{\underbrace{a_1}_{p}\underbrace{a_2}_{q}\underbrace{a_3}_{p}\underbrace{a_4}_{q}}) = \overline{\underbrace{a_4}_{q}\underbrace{a_3}_{p}\underbrace{a_2}_{q}\underbrace{a_1}_{p}} = (a_4 \times p^2 q + a_3 \times pq + a_2 \times p + a_1)_{10}$. After symbol-wise reversal, a p-q-alternate decimal number becomes a q-p-alternate decimal number. According to this type of index representation, we have the following theorems to maximize the girth of type-I cycles.

Theorem 1 *To maximize the girth of type-I cycles, connect auxiliary nodes in the lower tree T_L indexed by the q-p-alternate decimal x_{q-p} to auxiliary nodes in the upper tree T_U indexed by the p-q-alternate decimal $\pi_S(x_{q-p})$.*

Proof: From its definition, C_I contains four auxiliary nodes, say, a_1, a_2, a_3, and a_4, as shown in figure 6. Let the distance between a_1 and a_2 within the sub-tree T_U as $d_U(a_1, a_2)$, and the distance between a_3 and a_4 within the sub-tree T_L

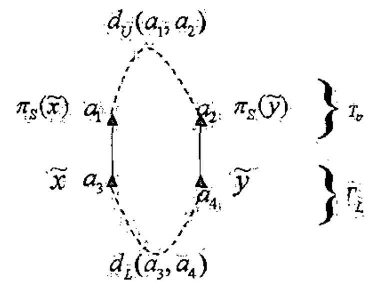

Fig. 6. Type-I cycle containing two T_U's auxiliary nodes a_1 and a_2 and two T_L's auxiliary nodes a_3 and a_4

by $d_L(a_3, a_4)$. The length of C_I is $d_U(a_1, a_2) + d_L(a_3, a_4) + 2$. The minimum value of $d_L(a_3, a_4)$ is 2 when a_3 and a_4 are both descendants of the same leaf-node of T_L. For this situation, to maximize the length of C_I, $d_U(a_1, a_2)$ should be as large as possible. However, for T_U of height h, the maximal value of $d_U(a_1, a_2)$ is $2h$. Therefore, the girth of C_I is less than or equal to $2h+4$. If we show, following the connecting rule in theorem 1, that all C_I formed have length at least $2h+4$, then theorem 1 is proved.

For T_L of height h, we need h symbols in the q-p alternate decimated labelling of all auxiliary nodes of T_L. Number the corresponding h coordinates of indices from 1 to h, starting from the rightmost coordinate. Similarly, h symbols are needed in the p-q-alternate decimated labelling of all T_U's auxiliary nodes and we number the corresponding h coordinates of their indices from 1 to h, also starting from the rightmost coordinate. The distance within T_L between two auxiliary nodes with indices $\widetilde{x}$ and $\widetilde{y}$ is $d_L(\widetilde{x}, \widetilde{y}) = 2k$, where k is the leftmost coordinate where the indices $\widetilde{x}$ and $\widetilde{y}$ differ from each other. This also applies to the distance within T_U between two auxiliary nodes with indices $\pi_S(\widetilde{x})$ and $\pi_S(\widetilde{y})$. After symbol-wise reversal, the $\widetilde{x}$'s and $\widetilde{y}$'s symbols at the coordinate k become $\pi_S(\widetilde{x})$'s and $\pi_S(\widetilde{y})$'s symbols at the coordinate $h+1-k$, respectively. So the symbols of $\pi_S(\widetilde{x})$ and $\pi_S(\widetilde{y})$ at the coordinate $h+1-k$ must also be different. Therefore,

$$d_U(\pi_S(\widetilde{x}), \pi_S(\widetilde{y})) \geq 2(h+1-k)$$

From figure 6, the length of such a C_I is then

$$L_C = d_L(\widetilde{x}, \widetilde{y}) + d_U(\pi_S(\widetilde{x}), \pi_S(\widetilde{y})) + 2 \geq 2h + 4$$

From the above analysis, all type-I cycles C_I that result from the construction in theorem 1 are at least of length $2h + 4$. This completes the proof.

We introduce now two operators $\dot{+}$ and $\dot{-}$. The operator $\dot{+}$ denotes the symbol-wise addition over the q-p-alternate decimal or p-q-alternate decimal while the operator $\dot{-}$ denotes the symbol-wise subtraction. Define the *symbol-wise shift* S as any p-q-alternate decimal number that has the same number of coordinates as $\pi_S(\widetilde{x})$. Theorem 1 is extended to theorem 2 using these concepts.

Theorem 2 *To maximize the girth of type-I cycles, connect the auxiliary nodes in T_L indexed by the q-p-alternate decimal x_{q-p} to the auxiliary nodes in T_U indexed by the p-q-alternate decimal $\pi_S(x_{q-p})\dot{+}(S)_{p-q}$.*

The proof of theorem 2 is similar to the proof of theorem 1. We do not provide the details. What theorem 2 says is that we are free to choose the value of $(S)_{p-q}$ in connecting x_{q-p} to $\pi_S(x_{q-p})\dot{+}(S)_{p-q}$. In the next step, we exploit this freedom in designing codes free of short type-II cycles.

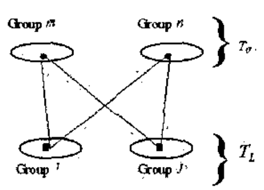

Fig. 7. Length 4 type-II cycle in turbo-like LDPC codes with column weight $j \geq 3$

Consider length 4 type-II cycles first. A length 4 type-II cycle is shown in figure 7. Here we need to introduce the concept of *group*. Auxiliary nodes in T_U belong to the same group if the symbols at the leftmost coordinate of their p-q-decimated indices are the same. T_U's leaf-nodes are in the same group if their descendants (auxiliary nodes) are in the same group. According to this definition, the leaf-nodes of T_U can be divided into $p = k-1$ groups. We notice that, the p groups correspond to p main branches of T_U. The auxiliary nodes and leaf-nodes of T_L can, likewise, be classified into $q \times p = (j-1)(k-1)$ different groups, which correspond to $(j-1)(k-1)$ main branches of T_L. The application of symbol-wise reversal guarantees that each leaf-node of T_U is connected to $q = j - 1$ groups out of totally qp groups in T_L. Let leaf-nodes of T_U in the same group be connected to the same $q = j-1$ groups in T_L and we collect the indices of the q groups in a set. Make such sets of T_U's leaf nodes in different groups to be distinct from each other, then length-4 type-II cycles are avoided.

To study length 6 type-II cycles, we need to apply theorem 2. When connecting any auxiliary node of T_L in group i to any auxiliary node of T_U in group j, we let the symbol-wise shift to be exactly the same, denoted as $S_{i,j}$. For different i, j, $S_{i,j}$ needs not to be the same. The problem is now how to choose appropriate $S_{i,j}$, for $i = 0, 1, \ldots, q \times p$ and $j = 0, 1, \ldots, p$. The following theorem helps to choose $S_{i,j}$ to eliminate length 6 type-II cycles:

Theorem 3 *When applying symbol wise reversal and shift $\pi_S(\widetilde{x})\dot{+}S_{i,j}$, a length 6 type-II cycle C_{II} exists if symbol-wise shifts $S_{i,j}$ satisfy one of the three conditions:*

(i) For three different groups i, j, and k in T_L and three different groups l, m, and n in T_U, one of the following equations is satisfied:

$$S_{i,l}\dot{+}S_{j,m}\dot{+}S_{k,n} = S_{i,m}\dot{+}S_{j,n}\dot{+}S_{k,l} \quad (1)$$
$$S_{i,l}\dot{+}S_{j,m}\dot{+}S_{k,n} = S_{i,n}\dot{+}S_{j,l}\dot{+}S_{k,m} \quad (2)$$
$$S_{i,m}\dot{+}S_{j,l}\dot{+}S_{k,n} = S_{i,l}\dot{+}S_{j,n}\dot{+}S_{k,m} \quad (3)$$
$$S_{i,m}\dot{+}S_{j,l}\dot{+}S_{k,n} = S_{i,n}\dot{+}S_{j,m}\dot{+}S_{k,l} \quad (4)$$
$$S_{i,n}\dot{+}S_{j,l}\dot{+}S_{k,m} = S_{i,m}\dot{+}S_{j,n}\dot{+}S_{k,l} \quad (5)$$
$$S_{i,n}\dot{+}S_{j,m}\dot{+}S_{k,l} = S_{i,l}\dot{+}S_{j,n}\dot{+}S_{k,m} \quad (6)$$

(ii) For two different groups i and j in T_L, and two different groups m and n in T_U,

$$0 < S_{i,n}\dot{-}S_{j,n}\dot{+}S_{j,m}\dot{-}S_{i,m} < p = k - 1 \quad (7)$$

(iii) For two different groups i and j in T_L, and two different groups m and n in T_U,

$$0 < \pi_S(S_{i,n}\dot{-}S_{j,n}\dot{+}S_{j,m}\dot{-}S_{i,m}) < qp = (j-1)(k-1) \quad (8)$$

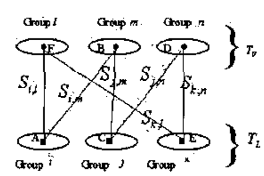

Fig. 8. Length 6 type-II cycle (Type(a))

Equations (1-6) correspond to the type (a) length 6 cycles shown in figure 8; inequality (7) and inequality (8) are derived from type (b) cycles and type (c) cycles shown in figure 9 respectively. Proof of theorem 3 is not provided here. For details, please refer to [11]. According to theorem 3, by choosing suitable symbol-wise shifts $S_{i,j}$ for $i = 0, 1, \cdots, k-1$ and $j = 0, 1, \cdots, (j-1)(k-1)$ that do not satisfy conditions (1-7), we can avoid all length 6

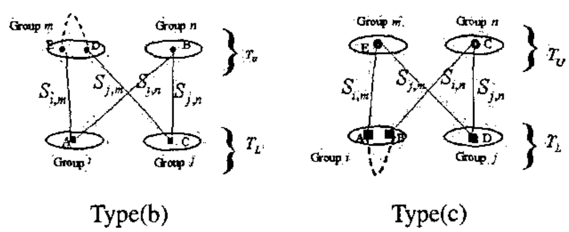

Fig. 9. Length 6 type-II cycle

type-II cycles while at the same time maximizing the girth of type-I cycles.

As an illustration, we applied the above methods to construct a (2457, 3, 9) regular LDPC code, rate 2/3, free of any cycles with length less than 8 and whose structure is shown by the 819×2457 parity-check matrix **H** shown in figure 10. In this matrix, along the solid lines there is a sin-

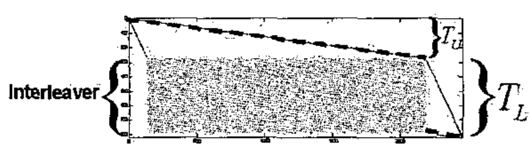

Fig. 10. Parity-check matrix of a (2457, 3, 9) LDPC code with girth 8

gle 1 in each row, while along the dashed thicker diagonals there are eight 1's in each row, so that per row there are nine 1's and in each column there are three 1's.

In particular, for column weight $j = 2$ turbo-like LDPC codes, we can further prove that the minimum length of type-II cycles is 8 and no $4k + 2, k = 2, 3, 4, \ldots$ (e.g., length 10 and length 14) type-II cycles exist. Again, Theorems 1 and 2 can be applied to maximize type-I cycles; Several theorems similar to theorem 3 are then employed to eliminate those length 8 and length 12 type-II cycles, hence forming a code with girth at least 16. For details, refer to [11].

4. CONCLUSION

In this paper, we propose a new class of well-structured LDPC codes—turbo-like LDPC codes. As mentioned, we can design large girth regular LDPC codes with flexible code rate in a turbo manner. Hence, large girth guarantees fast convergence of iterative decoding algorithms, large minimum distance between codewords, and alleviates error floor problems at high signal to noise ratio.

Another advantage of these codes is their reduced memory requirements. We need nj memory units to represent a randomly constructed (n, j, k) LDPC code when using look-up tables; look-up table has its drawback as it leads to a complicated memory-access architecture. In contrast, turbo-like LDPC codes can be described by a single $(j-1)(k-1) \times (k-1)$ matrix storing the symbol-wise shifts S_{ij}. For example, for the $(2457, 3, 9)$ turbo-like LDPC code constructed in this paper, instead of storing $3 \times 2457 = 7371$ row or column indices of those non-zero entries, we only need to store $(9-1)(3-1) \times (9-1) = 128$ entries of the symbol-wise shift matrix **S**, a significant reduction over look-up table methods.

5. 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 Transactions. on Information Theory*, vol. 45, no. 2, pp. 399-431, Mar. 1999.

[3] http://mars.bell-labs.com.

[4] Y. Li and J. Moon, "Increasing data rates through iterative coding and antenna diversity in OFDM-based wireless communication," appeared in *IEEE Globecom'01*, San Antonio, TX.

[5] 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.

[6] R. Michael Tanner, Deepak Sridhara, and Tom Fuja, "A class of group-structured LDPC codes" to appear in *ICSTA 2001 Proceedings*, Ambleside, UK.

[7] 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.

[8] T. Richardson, A. Shokrollahi, and R.Urbanke, "Design of capacity-approaching irregular low-density parity check codes," *IEEE Trans. on Information Theory*, vol. 47, no. 2, Feb. 2001, pp. 619-637.

[9] R. L. Townsend and E.J. Weldon, "Self-orthogonal quasi-cyclic code," *IEEE Transactions on Information Theory*, vol. IT-13, no.2, pp. 183-195, April 1967.

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

[11] Jin Lu, José M. F. Moura, "Turbo-like LDPC codes," submitted, March 2003.