Low-Floor Tanner Codes Via Hamming-Node or RSCC-Node Doping[⋆]

Shadi Abu-Surra[1], Gianluigi Liva[2], and William E. Ryan[1]

[1] Electrical and Computer Engineering Department, University of Arizona
ryan@ece.arizona.edu, shadia@ece.arizona.edu
[2] Dipartimento di Elettronica, Informatica e Sistemistica, Universita di Bologna
gliva@deis.unibo.it

Abstract. We study the design of structured Tanner codes with low error-rate floors on the AWGN channel. The design technique involves the "doping" of standard LDPC (proto-)graphs, by which we mean Hamming or recursive systematic convolutional (RSC) code constraints are used together with single-parity-check (SPC) constraints to construct a code's protograph. We show that the doping of a "good" graph with Hamming or RSC codes is a pragmatic approach that frequently results in a code with a good threshold and very low error-rate floor. We focus on low-rate Tanner codes, in part because the design of low-rate, low-floor LDPC codes is particularly difficult. Lastly, we perform a simple complexity analysis of our Tanner codes and examine the performance of lower-complexity, suboptimal Hamming-node decoders.

1 Introduction

An LDPC code, as first proposed by Gallager in [1], is defined as an (n, k) linear block code with a low density of non-zero elements in its parity check matrix H. The $m \times n$ matrix H can be represented as a bipartite graph (Tanner graph) with n variable nodes and m single-parity-check (SPC) nodes. A generalization of these codes was suggested by Tanner in [2], for which subsets of the variable nodes obey a more complex constraint than an SPC constraint, such as a Hamming code constraint. There are at least two advantages to employing constraint nodes with constraints more complex than a simple parity check. First, more complex constraints tend to lead to larger minimum distances. Second, because a complex constraint node can encapsulate multiple SPC constraints, the resulting Tanner graph will contain fewer edges so that deleterious graphical properties are more easily avoided. Both of these advantages lead to a lower error-rate floor. One successful instance of a Tanner code is the turbo product code (TPC) [3]. Another special case of Tanner codes was studied in [4] and [5], where the constraint nodes correspond to Hamming codes. Also, in [6] codes are built by applying BCH or Reed-Solomon code constraints to variable node

[⋆] This work was funded by NASA-JPL grant 1264726 and by the University of Bologna, Progetto Pluriennale.

subsets, and in [7] recursive systematic convolutional (RSC) codes are used as constraints. The RSC-LDPC codes in this work are more general in the sense that different constraint nodes can be used to construct codes and the graph structure is generally more flexible.

Liva and Ryan in [8], [9] present a more general case of Tanner codes in [5] called Hamming-doped LDPC codes (HD-LDPCC). This generalization allows more than one type of constraint node in the graph as well as irregularity among the node degrees. The doping refers to the fact that the design approach involves inserting Hamming constraint nodes into a Tanner graph or a protograph [10] in place of selected SPC nodes. (A protograph will be defined in Section III.) In this paper, we consider the doping of protographs using either Hamming nodes or RSC nodes; we will call the latter code type RSC-LDPC codes. When referring generically to such a code, we will use doped LPDC code and Tanner code interchangeably. We will also refer to a code that resides at a constraint node as a component code (in contrast with Tanner's "subcode"), and we use constraint node and component-code node interchangeably.

We demonstrate via computer simulations that both HD-LDPC and RSC-LPDC codes exhibit very low error floors, even for code lengths less than 1000 bits. Of course, since our doping technique replaces SPC nodes of code rate $p/(p+1)$ by lower-rate codes, the resulting doped LDPC codes are low-rate codes. Thus, our code design technique provides an approach to designing structured, short (or long), low-rate graphical codes with very low floors, a difficult task if one were restricted to standard LDPC codes [11].

The paper proceeds as follows. In the next section, we present an overview of the construction of Hamming- and RSCC-doped LDPC codes based on protographs. Section III presents four example code family designs. In Section 4, we discuss the iterative decoders which are used to decoder the doped LDPC codes, and analyze their complexity. In Section 5, we present simulation results of the codes we have designed.

2 Overview of the Design Technique

The graph of a Tanner code has n variable nodes and m_c constraint nodes. The connections between the set of variable nodes and constraint nodes V and C is given by an $m_c \times n$ adjacency matrix Γ. For an LDPC code, the adjacency matrix Γ and the parity-check matrix H are identical. For a Tanner code, knowledge of the parity-check matrices of the component codes is also required.

In this paper, we consider only Hamming or RSC component codes in addition to the more common SPC component codes. Because the parity-check matrices for SPC and Hamming codes are straightforward, we discuss only the parity-check matrices for (possibly punctured) rate-1/2 finite-length RSC codes which will be used to dope graphs. For a memory-ν, rate-1/2 RSC code with generator polynomials $g_1(D) = g_{10} + g_{11}D + \cdots + g_{1\nu}D^{\nu}$ and $g_2(D) = g_{20} + g_{21}D + \cdots + g_{2\nu}D^{\nu}$, the corresponding parity-check matrix is

$$H(D) = \left[g_2(D) \; g_1(D) \right] . \quad (1)$$

Because we consider finite block lengths, the binary parity-check matrix for such a code is given by

$$H = \begin{bmatrix} g_{20} & g_{10} & 0 & 0 & 0 & 0 & \cdots \\ g_{20} & g_{10} & g_{20} & g_{10} & 0 & 0 & \cdots \\ \vdots & \vdots & g_{20} & g_{10} & & & \\ g_{2\nu} & g_{1\nu} & \vdots & \vdots & & & \\ 0 & 0 & g_{2\nu} & g_{1\nu} & & & \\ 0 & 0 & 0 & 0 & & \ddots & \\ \vdots & \vdots & \vdots & \vdots & & & \end{bmatrix}, \tag{2}$$

To find the rate of a doped graph with n variable nodes and m_c constraint nodes, note that each component-code contributes $(1 - R_i)n_i$ redundant bits, where n_i and R_i are the length and rate of the i^{th} component-code, respectively. Consequently, the total number of redundant bits in the code cannot exceed $m = \sum_{i=1}^{m_c}(1 - R_i)n_i$, and so the number of information bits in the code will be at least $n - m$. This implies that the code rate satisfies $R_c \geq 1 - \frac{m}{n}$, with equality when the check equations are independent.

The parameters in standard LDPC code design which most affect code performance are the degree distributions of the node types, the topology of the graph (e.g., to maximize girth), and the minimum distance, d_{min}. For the design of Tanner codes, decisions must also be made on the types and multiplicities of component codes to be used. The choice of component code types and their multiplicities is dictated by the code rate and complexity requirements. Regarding complexity, we consider only Hamming codes for which the number of parity bits is $(1 - R_i)n_i \leq 4$ and only RSC codes for which the number of trellis states is at most eight. Note that this constraint on the Hamming code family limits the number of states in the time-varying BCJR trellis [12] to be at most 16.

As for LDPC codes, the topology of the graph for a Tanner code should be free of short cycles. Obtaining optimal or near-optimal degree distributions for the graphs of Tanner codes can proceed as for LDPC codes, using EXIT charts [13], for example. In this paper, we instead follow the pragmatic design approach introduced in [8], [9], which starts with a protograph that is known to have a good decoding threshold and replaces selected SPC nodes with either Hamming or RSC nodes. Although we provide no proof, the substitution of these more complex nodes tends to increase minimum distance as shown by simulations. Further, it leads to a smaller adjacency matrix since multiple SPC nodes are replaced by a single component code node. The implication of a smaller adjacency matrix is that short cycles and other deleterious graphical properties are more easily avoided.

3 Example Doped LDPC Code Designs

A protograph [14], [10] is a relatively small bipartite graph from which a larger graph can be obtained by a copy-and-permute procedure: the protograph is copied q times, and then the edges of the individual replicas are permuted among the replicas (under restrictions described below) to obtain a single, large graph. Of course, the edge connections are specified by the adjacency matrix Γ.

Note that the edge permutations cannot be arbitrary. In particular, the nodes of the protograph are labeled so that if variable node A is connected to constraint node B in the protograph, then variable node A in a replica can only connect to one of the q replicated B constraint nodes. Doing so preserves the decoding threshold properties of the protograph. A protograph can possess parallel edges, i.e., two nodes can be connected by more than one edge. The copy-and-permute procedure must eliminate such parallel connections in order to obtain a derived graph appropriate for a parity-check matrix.

It is convenient to choose an adjacency matrix Γ as an $M_c \times n_c$ array of $q \times q$ weight-one circulant matrices (some of which may be the $q \times q$ zero matrix). We will call each row of permutation matrices a *block row* which we observe has q rows and $n = qn_c$ columns. We note that there is one block row for each constraint node of the protograph. We note also that the number of nonzero permutation matrices in a block row is simultaneously equal to the degree of its corresponding constraint nodes and the common length of the nodes' component codes.

Since there is one matrix H_i for each block row of Γ (for the i^{th} component code), we need only discuss the i^{th} block row. Let H_i be $m_i \times n_i$. Then for each row in the i^{th} block row, replace the n_i ones in the row by the corresponding n_i columns of H_i. This expands the i^{th} block row from $q \times n$ to $qm_i \times n$. (For the special case of an SPC constraint node, $m_i = 1$ and the row block is not expanded.) Once this process has been applied to each block row, the resulting parity-check matrix H for the Tanner code will be $\sum_i qm_i \times n$. Because Γ is block circulant, the resulting matrix H can also be put in a block-circulant form (thus, the Tanner code will be quasi-cyclic) [9].

For the case when Γ is not an array of circulants, the H matrix can be obtained via a process analogous to the one above. Γ in this case corresponds to a random permutation on the edges of the protograph replicas, but two constraints are taken in considerations: the protograph structure and the girth of the graph.

In the remainder of this section, we present several HD-LDPC and RSC-LDPC codes whose design relies on doping protographs. In Section 5 we present selected simulation results for these codes on the AWGN channel.

Code 1: Rate-1/6 HD-LDPC Code. The doped protograph for a rate-1/6 HD-LDPC code is shown in Figure 1. The protograph displays a single information bit, u_0, five parity bits p_0 to p_4, two SPC nodes, and a (6,3) shortened Hamming code. The initial protograph that we doped was a rate-1/4 ARA protograph [15], but with minor modification.

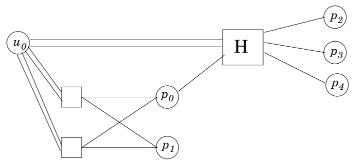

Fig. 1. Rate-1/6 HD-LDPC protograph

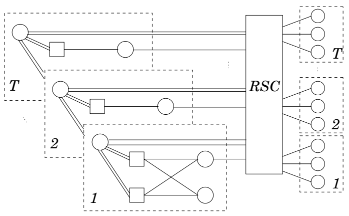

Fig. 2. Rate-1/6 RSC-LDPC protograph

The (6,3) Hamming code was selected because it leads to the targeted rate of 1/6, it has a low-complexity BCJR decoder, and its H matrix-based graph is free of 4-cycles so that belief propagation is an option. Note also that the addition of the Hamming node has the effect of amplifying the minimum distance of the eventual code (after copying and permuting). This is because there will be q copies of the Hamming node whose codewords have a minimum distance of three. Section 5 presents an example code based on this protograph together with its performance (a pseudo-random adjacency matrix is used).

Code 2: Rate-1/6 RSC-LDPC Code. The idea of adding a component code node to amplify weight (hence, d_{min}) led us to consider RSC nodes, particularly since RSC codes produce large weight for low-weight inputs. Since a rate-1/2 RSC code can have any even length, we must consider in the design of an RSC-doped protograph what this length should be. Figure 2 accommodates an unterminated ($6T$, $3T$) RSC component code, where T is a design parameter, so that the overall protograph has T inputs and $6T$ outputs. The $6T$ outputs are represented by all of the circles in Figure 2, some of which are obscured; the RSC node in Figure 2 has $3T$ inputs and $3T$ outputs. Notice that this figure contains T equivalent *sub-protographs.* In the copy-and-permute procedure, we ignore the fact that these were formerly protographs, and apply the copy-and-permute rules only to the overall protograph.

We point out that codes based on this protograph are turbo-like [16] in the sense that copies of the information bits are permuted and distributed over T accumulators, and then part of their outputs together with the remaining information bits are permuted and fed to the RSC code encoder. One major difference, however, is that the present code uses multiple short RSC code blocks rather than one or two long RSC code blocks. The rate-1/6 RSC-LDPC codes presented in Section 5 utilize (pseudo-)random adjacency matrices.

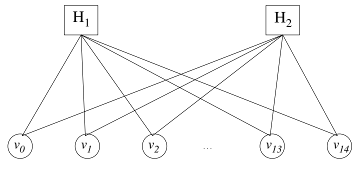

Fig. 3. Rate-1/2 HD-LDPC protograph

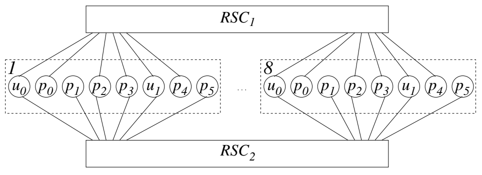

Fig. 4. Rate-1/4 RSC-LDPC protograph

Code 3: Rate-1/2 HD-LDPC Code. The protograph in Figure 3 corresponds to a rate-1/2 HD-LDPC code. It consists of two (15,11) Hamming component codes and 15 variable nodes. One of the protograph's variable nodes is punctured to achieve the desired rate. Further, the two code-components are not identical. Specifically,

$$H_1 = [M_1 \; M_2] = \begin{bmatrix} 1\,0\,1\,0\,1\,0\,1\,0 & 1\,0\,1\,0\,1\,0\,1 \\ 0\,1\,1\,0\,0\,1\,1\,0 & 0\,1\,1\,0\,0\,1\,1 \\ 0\,0\,0\,1\,1\,1\,1\,0 & 0\,0\,0\,1\,1\,1\,1 \\ 0\,0\,0\,0\,0\,0\,0\,1 & 1\,1\,1\,1\,1\,1\,1 \end{bmatrix}, \tag{3}$$

and $H_2 = [M_2 \; M_1]$, where the definitions of M_1 and M_2 are evident. The benefit of permuting the bits of identical component codes was pointed out by Tanner [2].

A rate-1/2 (2044,1022) Tanner code can be constructed from the protograph of Figure 3 as follows. First, make $q = 146$ total replicas of the protograph. This yield a graph with $n = (15)(146) = 2190$ bit nodes and $m_c = 292$ check nodes. The number of parity bits for the code is $m = 292(15-11) = 1168$ so that the resulting code is (2190,1022). For the code presented in Section 5, Γ is an array of $q \times q$ circulants, in which case, the code quasi-cyclic. A rate-1/2 (2044,1022) quasi-cyclic Tanner code can be obtained by puncturing the first 146 bits of each codeword (corresponding to the first column of circulants of Γ).

Code 4: Rate 1/4 RSC-LDPC Code. As depicted in Figure 4, we can obtain a rate-1/4 RSC-LDPC protograph which resembles the protograph of Figure 3, with two different rate-1/2 RSC nodes (of length 48) used in place of the Hamming nodes. Note that the two RSC component-codes form 48 parity check equations, which necessitate the existence of 64 variable nodes in the protograph in order to achieve a rate-1/4 code. Moreover, the number of information bits among these 64 bits is 16 and each 64-bit word must satisfy these 48 check equations. In Figure 4, we divided the variable nodes into eight similar groups (enclosed in the dash boxes), with six connections to each RSC code. Each group contains two information bits, u_0 and u_1, and six parity bits, p_0 to p_5, which are ordered in a sequence relevant to the decoder.

The rate-1/2 RSC component codes have two different polynomial sets; one has polynomials $(17, 15)_8$ and the other has polynomials $(3, 2)_8$. Assuming that both have unterminated trellises; the resultant code has rate 1/4. However, we have to terminate one of the two component codes to obtain good performance. (Terminating both of them also works, but at the cost of code rate.) In this code, the $(17, 15)_8$ RSC code trellis has been terminated. Since $\nu = 3$ and the rate is 1/2 for this component code, 6 code bits are related to these termination bits.

From Figure 4, the last six bits of each of the RSC component codes include two information bits. Consequently, trellis termination process reduces the rate from 16/64 to 14/64. In order to obtain rate 1/4, we puncture eight of the 64 bits, four degree-one variable nodes from each RSC code.

Finally, we constructed a (16352,4088) RSC-LDPC code by making 292 copies of the above protograph. A block-circulant adjacency matrix was used in our simulations. In summary, $n = 18\,688$, $m_c = 584$, $M_c = 2$, and $q = 292$.

4 Doped-LDPC Code Iterative Decoder

For LDPC codes in this paper, we used the standard sum-product algorithm (SPA). For the Tanner codes which have more complex constraint nodes, a soft-input soft-output (SISO) decoder is used to compute the soft-output messages. The choice of the SISO decoder for non-SPC constraint codes depends on the code type. For RSC codes we use the BCJR decoder [17].

In HD-LDPC codes, the Hamming constraints can be replaced by their equivalent SPC equations. However, except for the (6,3) shortened Hamming code, the large number of 4-cycles the resultant graph degrades the performance of the SPA decoder. Alternatively, for the Hamming nodes, we can use the BCJR decoder applied to the BCJR trellis [12]. We also consider the modified Chase algorithm [18] and the cyclic-2 pseudo-maximum likelihood (PML) decoder [19].

The modified Chase and cyclic-2 PML decoders are both SISO list-based decoders. Cyclic-2 has an advantage over modified Chase in term of complexity as it uses a list that refers to nearby codewords, which are independent of its input, resulting in fewer addition operations. The complexity reduction factor from using either of these decoders instead of the BCJR decoder depends on the number of the states in the code's trellis. As an example, in the decoding of 10^7 codewords of the (32, 26) extended Hamming code, we observed that the cyclic-2 decoder was 9 times faster than BCJR decoder, and the modified Chase decoder was 4.5 times faster than BCJR.

Lastly, to gain insight on the decoding complexity of the HD-LDPC and RSC-LDPC codes compared with that of standard regular LDPC we consider the following rate 1/6 codes. The first is an HD-LDPC code constructed from W copies of the protograph in Figure 1. The second is an RSC-LDPC code based on one copy of the protograph in Figure 2, using the RSC code polynomials $(5, 7)_8$. The last code is an LDPC code derived from the previous HD-LDPC code, where the Hamming constraint is replaced by its SPC constraints.

The number of additions per iteration are $131W$, $50W$, and $20W$ for HD-LDPC, RSC-LDPC, and LDPC codes, respectively. This calculation is based on the following (η is the relevant block length, $N_{s,total}$ is the total number of trellis states in the finite-length trellis): (1) For a standard LDPC codes, the number of additions equals to the number of ones in its parity-check matrix. (2) The number of additions in the HD-LDPC BCJR is given by $2N_{s,total} + 4\eta$. (3) For the RSC-LDPC BCJR, the number of additions is $2N_{s,total} + 5\eta/2$ because the number of stages in a rate-1/2 RSC trellis is half the block length, but it has to compute two values at each stage; hence, 5 instead of 4.

5 Simulation Results

In this section we present several simulation results for different doped-LDPC codes. First, we designed a (600, 100) HD-LDPC code based on the protograph in Figure 1 and three (600, 100) RSC-LDPC codes based on the protograph in Figure 2. The three RSC-LDPC codes correspond to three different values of the parameter T: $T = 2$, 4, and 8. All of these codes were constructed using random permutations on the edges of their protographs, but two constraints are taken into consideration: the protograph structure and the girth of the graph. The progressive edge growth construction in [20] is used to give the required girth, which is eight for all loops that have only SPC nodes. On the other hand, loops that include Hamming or RSC nodes can be of length less than eight.

A comparison between the frame error rate (FER) curves of these codes and the (600,100) random coding bound (RCB) is presented in Figure 5. The iterative decoder described above was used, where BCJR decoders are used to decode the Hamming and RSC component codes. The maximum number of iterations is $I_{max} = 50$ and 20 error events were collected at each E_b/N_0 value on each curve, except for the point at 4.5 dB of the $T = 8$ RSC-LDPC curve where only three error events occurred during the decoding of 7.26×10^8 codewords. Note that the floor for almost every code is quite low, even though the code length is 600. Note also the lowest floor occurs for the $T = 8$ RSC-LDPC code, which shows no evidence of a floor down to FER $\approx 10^{-9}$. This code is about 1.3 dB from the random coding bound at FER$=10^{-4}$.

Figure 6 shows the error rate performance curves of the (2044, 1022) quasi-cyclic HD-LDPC code. The Hamming component codes were decoded using the BCJR decoder and the overall decoder employed a maximum of $I_{max} = 50$ iterations. The code performance is within 1 dB of the random coding bound and has no floor down to FER $\approx 5 \times 10^{-8}$.

The performance of the rate-1/4 RSC-LDPC code ($I_{max} = 20$) constructed in Section 2 is presented in Figure 7. Its performance is compared to that of

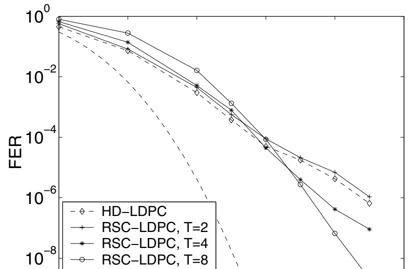

Fig. 5. Frame error rate comparison between (600, 100) HD-LDPC code and RSC-LDPC codes at different T, $I_{max} = 50$

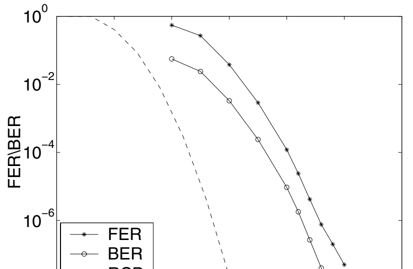

Fig. 6. Performance of (2044, 1022) HD-LDPC code compared to the random coding bound. $I_{max} = 50$.

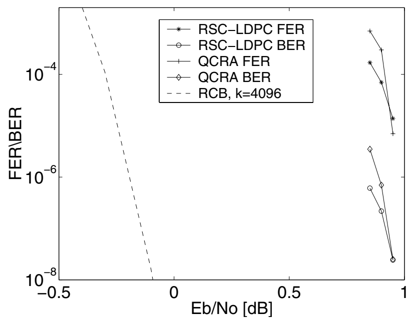

Fig. 7. Performance of (16352, 4088) RSC-LDPC code compared to that of (16384, 4096) QCRA code. $I_{max} = 20$.

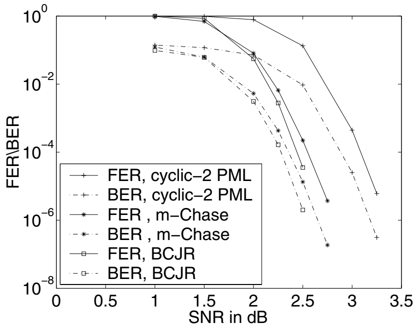

Fig. 8. Comparison between the performance of BCJR decoder, and the other sub-optimal decoders. The (2048, 1024) HD-LDPC components are (32, 26) extended Hamming codes.

the quasi-cyclic repeat-accumulate code (QCRA) in [21] as well as the random coding bound. The curves show that our code is superior to the QCRA code at low E_b/N_0 values. But at higher E_b/N_0 values, the QCRA code has a slightly better FER than the RSC-LDPC. We noticed that by increasing I_{max} from 20 to 50 in RSC-LDPC code, the FER at $E_b/N_0 = 0.8$ dB reduced to around 2×10^{-6}.

Finally, we examined the performance of a (2048, 1024) HD-LDPC code, constructed from the (32, 26) extended Hamming code, using the BCJR decoder, the Chase decoder (radius 6), and the cyclic-2 PML decoder. Note in Figure 8 that the performance curves of the modified Chase and the BCJR decoders are almost the same, and about 0.5 dB better than that of the cyclic-2 PML decoder. On the other hand, cyclic-2 PML decoder is about twice as fast as the Chase decoder and about nine times as fast as the BCJR decoder.

Acknowledgments

We would like to acknowledge Professor Keith Chugg of the University of Southern California and Professor Marc Fossorier of the University of Hawaii for helpful discussions on finite code length performance bounds, and Prof. Chugg for his program which computes such bounds.

References

1. R. G. Gallager, "Low-density parity-check codes," *IRE Transactions on Information Theory*, vol. 8, pp. 21–28, January 1962.
2. R. M. Tanner, "A recursive approach to low complexity codes," *IEEE Transactions on Information Theory*, vol. 27, pp. 533–547, September 1981.
3. P. Elias, "Error free coding," *IRE Transactions on Information Theory*, vol. PGIT-4, pp. 29–37, September 1954.

4. M. Lentmaier and K. S. Zigangirov, "Iterative decoding of generalized low-density parity-check codes," in *IEEE International Symposium on Information Theory*, p. 149, August 1998.
5. J. Boutros, O. Pothier, and G. Zemor, "Generalized low density (Tanner) codes," in *IEEE International Conference on Communications, ICC '99*, pp. 441–445, June 1999.
6. N. Miladinovic and M. Fossorier, "Generalized LDPC codes with Reed-Solomon and BCH codes as component codes for binary channels," in *IEEE Global Telecommunications Conference, GLOBECOM '05*, November 2005.
7. S. Vialle and J. Boutros, "A Gallager-Tanner construction based on convolutional codes," in *Proceedings of International Workshop on Coding and Cryptography, WCC'99*, pp. 393–404, January 1999.
8. G. Liva and W. E. Ryan, "Short low-error-floor Tanner codes with Hamming nodes," in *IEEE Military Communications Conference, MILCOM '05*, 2005.
9. G. Liva, W. E. Ryan, and M. Chiani, "Design of quasi-cyclic Tanner codes with low error floors," in *4th International Symposium on Turbo Codes, ISTC-2006*, April 2006.
10. J. Thorpe, "Low-density parity-check (LDPC) codes constructed from protographs," Tech. Rep. 42-154, IPN Progress Report, August 2003.
11. Y. Mao and A. H. Banihashemi, "A heuristic search for good low-density parity-check codes at short block lengths," in *IEEE International Conference on Communications, ICC 2001*, pp. 41–44, June.
12. R. J. McEliece, "On the BCJR trellis for linear block codes," *IEEE Transactions on Information Theory*, vol. 42, pp. 1072–1092, July 1996.
13. S. ten Brink, G. Kramer, and A. Ashikhmin, "Design of low-density parity-check codes for modulation and detection," *IEEE Transactions on Communications*, vol. 52, pp. 670–678, April 2004.
14. I. D. S. Lin, J. Xu and H. Tang, "Hybrid construction of LDPC codes," in *Proc. of the 40th Annual Allerton Conference on Communication, Control, and Computing, Illinois*, October 2002.
15. A. Abbasfar, D. Divsalar, and K. Yao, "Accumulate repeat accumulate codes," in *IEEE Global Telecommunications Conference, GLOBECOM '04*, pp. 509–513, November 2004.
16. D. Divsalar, H. Jin, and R. McEliece, "Coding theorems for "turbo-like" codes," in *Proc. of 36th Allerton Conf.*, September 1998.
17. L. R. Bahl, J. cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Transactions on Information Theory*, vol. 20, pp. 284–287, March 1974.
18. R. M. Pyndiah, "Near-optimum decoding of product codes: Block turbo codes," *IEEE Transactions on Communications*, vol. 46, pp. 1003–1010, August 1998.
19. "Block decoding with soft output information," Patent 5930272, United States Patent, July 1999.
20. X. Y. Hu, E. Eleftheriou, and D. M. Arnold, "Progressive edge-growth Tanner graphs," in *IEEE Global Telecommunications Conference, GLOBECOM '01*, pp. 995–1001, November 2001.
21. R. M. Tanner, "On quasi-cyclic repeat-accumulate codes," in *Proc. of the 37th Annual Allerton Conference on Communication, Control, and Computing, Monticello, Illinois*, September 1999.