Turbo-Gallager Codes: The Emergence of an Intelligent Coding Scheme

P. C. Catherine

K. M. S Soyjaudah

Department of Electrical and Electronics Engineering
University of Mauritius

c.catherine@uom.ac.mu, ssoyjaudah@uom.ac.mu

ABSTRACT

In 1948, C. Shannon developed fundamental limits on the efficiency of communication over noisy channels. However it is only in 1993 (about half a century later) that Berrou, Glavieux and Thitimajshima developed turbo codes and demonstrated performance close to that limit. Overnight, much of the algebraic coding techniques of the pre-turbo era were rendered obsolete. Turbo codes employ iterative algorithms and focus on exchange of soft (or probabilistic) information. Shortly after, it was recognised that another class of codes developed by Gallager in 1963 shared all the essential features of turbo codes, including sparse graph-based codes and iterative message-passing decoders. Today, both turbo codes and low-density parity-check codes are largely superior to other code families and are being used in an increasing number of modern communication systems including 3G standards, satellite and deep space communications. However, the two codes have certain distinctive characteristics that differentiate them. This work proposes a blend of the two technologies, yielding a code that we nicknamed Turbo-Gallager or TG Code. The code has additional "intelligence" compared to its parents. It detects and corrects the so-called "undetected errors" and recovers from individual decoder failure by making use of a network of decoders.

Keywords: TG codes, message-passing algorithms, low-density parity-check codes, turbo codes, bipartite graphs

1. INTRODUCTION

The fundamental limit of communication over noisy channels, as set by Shannon (1948) in his seminal paper has remained an unattainable goal for generations of coding theorists. Indeed, Shannon proved his results mathematically and left no tangible clue as to the elaboration of practical codes capable of achieving this limit. For a long time, the problem was even thought as insoluble, until Berrou et al. (1993) presented their turbo codes. Their result marked the beginning of a renewed interest in the field.

In 1997, i.e. only four years after, the Jet Propulsion Laboratory (JPL), which carries out research for NASA, used turbo codes in the Pathfinder mission to transmit photographs of the Martian surface back to Earth (Burr 2001). More recently, in 2003, the European Space Agency (ESA) launched the first satellite (SMART-1) using turbo code technology. According to a report from the Consultative Committee for Space Data Systems (CCSDS), SMART-1 could be used in the future to enable the reliable transfer of large amounts of scientific data that researchers hope will shed light on the origins of the Moon (CCSDS 2004). It is also worthwhile noting that turbo-codes have been integrated in the Universal Mobile Telecommunications System (UMTS) 3G mobile radio standard (ETSI 2000).

However turbo codes are not the only contender to approach the Shannon limit. Back in the 1960's, Gallager in his PhD thesis worked on low-density parity-check (LDPC) codes (Gallager 1963). However, computational power at that time could not demonstrate the real capability of the code. Some thirty years later, MacKay & Neal (1995) from Cambridge, rediscovered LDPC codes and demonstrated their near-Shannon limit achieving capacity. From then on, turbo codes had a serious challenger. In 2002 for example, the Digital Video Broadcasting (DVB) Project initiated the search for a second generation standard for broadband satellite applications: DVB-S2. LDPC codes eventually beat six turbo codes to become the new standard (ETSI 2005). In addition LDPC codes have also been proposed for IEEE 802.11n (WiFi) and IEEE 802.16e (WiMAX) wireless communication standards as well as the IEEE 802.3 (10GBASE-T Ethernet). Recently, in August 2006, CCSDS has also published an experimental specification for use of LDPC codes in near-earth and deep space applications (CCSDS 2006). As things stand, LDPC codes and turbo codes will in a near future dominate the market of forward error correction. Already thousands of technical papers have been published on these two codes rendering much of the preceding 50 years of coding theory obsolete.

The success of the two codes boils down to certain common features they share. For example, the necessary "randomness" ingredient for good performance (Shannon 1948) is present in both. In turbo codes a random interleaver is used to permute data between two decoders whereas a random technique is usually employed to create the parity-check matrix of an LDPC code. Both codes are decoded using message-passing algorithms (Kschischang et al. 2001). Soft Output Viterbi Algorithm (SOVA) (Hagenauer & Hoeher 1989) or Maximum Aposteriori (MAP) (L. Bahl & Raviv 1974) algorithms are used for turbo codes whereas the sum-product algorithm (MacKay 1999) is used for LDPC codes.

However, both codes are also characterized by the same weakness, the error floor phenomenon (Valenti & Woerner 1997, Richardson 2003) which is a direct effect of the *monotonic* behavior of the decoding algorithms. Once the decoder has settled in a given direction, whether right or wrong, it will carry on in that direction. This characteristic of message passing algorithms may lead to 1) undetected errors, i.e. the decoder trajectory converges to a valid codeword different from the transmitted one, 2) near-codewords, a state of low *entropy* where the decoder gets stuck and cannot recover, or 3) continuous oscillations between certain states, with no definite convergence. In this work, we shall use LDPC codes in a modified turbo concatenating structure (nicknamed Turbo-Gallager or TG codes) and devise an appropriate decoding algorithm that allows for non-monotonic behavior. To the best of our knowledge, *no* other codes share this faculty. Indeed TG codes (Soyjaudah & Catherine 2002) have in-built capacity to take "intelligent decisions".

With a view of interesting a wider audience, mathematical underpinnings of the work and exact formulation of algorithms have been dropped in favor of a more generic approach. The paper is organized as follows. In section 2, we provide a brief description of turbo codes and LDPC codes. We introduce the encoding and decoding scheme of TG codes in section 3. Performance results and analysis are presented in section 4, and we conclude with section 5.

2. BRIEF OVERVIEW OF TURBO AND LDPC CODES

In this section, we provide a brief description of the encoding and decoding process of both turbo and LDPC codes. For an in-depth treatment, please refer to (Berrou et al. 1993, Valenti 1999) for turbo codes and (MacKay 1999) for LDPC codes.

2.1 Turbo Codes

Figure 1(a) depicts a schematic representation of the encoding process of turbo codes. Note that Berrou et al. (1993) used two Recursive Systematic Convolutional (RSC) encoders in their work. However, the *turbo principle* remains essentially the same for other encoders. A message $\mathbf{u}$ of size k bits is used as input to a first encoder. The same message is then interleaved (scrambled) and fed to a second encoder. Finally, the resulting parity bits P_1 of size n_1 and P_2 of size n_2 are appended to $\mathbf{u}$ to form the codeword (see Figure 1(a). Note that the resulting *rate* of the code is given by $k/\left(k+n_1+n_2\right)$. We characterize the channel by Additive White Gaussian Noise (AWGN) (see Figure 1(b)). At the receiving side, the concatenated sequence $\mathrm{R_u}, \mathrm{R_{P1}}$ and $\mathrm{R_{P2}}$ corresponding to the noisy received version of $\mathbf{u}$, P_1 and P_2 are fed to two decoders as described in Figure 1(c). The innovation of the approach is the *sharing* of information between the two decoders. This is probably the most powerful aspect of turbo codes, and alas its greatest weakness. The decoders work in a *cooperative* fashion, and as such cannot contradict each other, leading to a monotonic behavior, which at times can be fatal.

2.2 Low-Density Parity-Check Codes

An (n, k) LDPC code of rate k/n maps a source sequence $\mathbf{s}$ of k bits to the transmitted codeword $\mathbf{t}$ of n bits through the linear operation $\mathbf{t} = \mathbf{G^T s}$. $\mathbf{G^T}$ is the generator-transpose matrix of the code and is related to a *sparse* parity-check matrix $\mathbf{H}$ such that $\mathbf{HG^T} = \mathbf{0}$. The decoding of LDPC codes is most efficiently described over the *bipartite* graph representing the code (Tanner 1981, MacKay 1999, Kschischang et al. 2001). A bipartite graph is one in which the nodes can be partitioned into two disjoint classes. An edge of the graph may connect a node of one class to a node of the other class, but there are no edges connecting nodes of the same class (Tanner 1981). Figure 2 shows an example the parity-check matrix of a (16,4) LDPC code. Its associated bipartite graph is shown in Figure 3. During the decoding process, check nodes and bit nodes (respectively represented by the black boxes and empty circles of Figure 3) exchange information. One interesting issue is that the sum-product algorithm provides posterior probabilities over the states of the parity bits. For turbo codes such probabilities are unavailable. Once the two sets of nodes have exchanged information, we say the algorithm has run for one iteration. If a valid codeword is detected, we stop the algorithm and declare success. Else, decoding continues until some maximum preset number of iterations is reached, at which time failure is declared. Note that the decoding algorithm acts locally, and as such is subject to the influence of potentially “bad nodes”. These nodes propagate wrong beliefs throughout the bipartite graph and induces their neighbors to alter their states accordingly. As such, the decoder may get trapped, resulting in decoding failure.

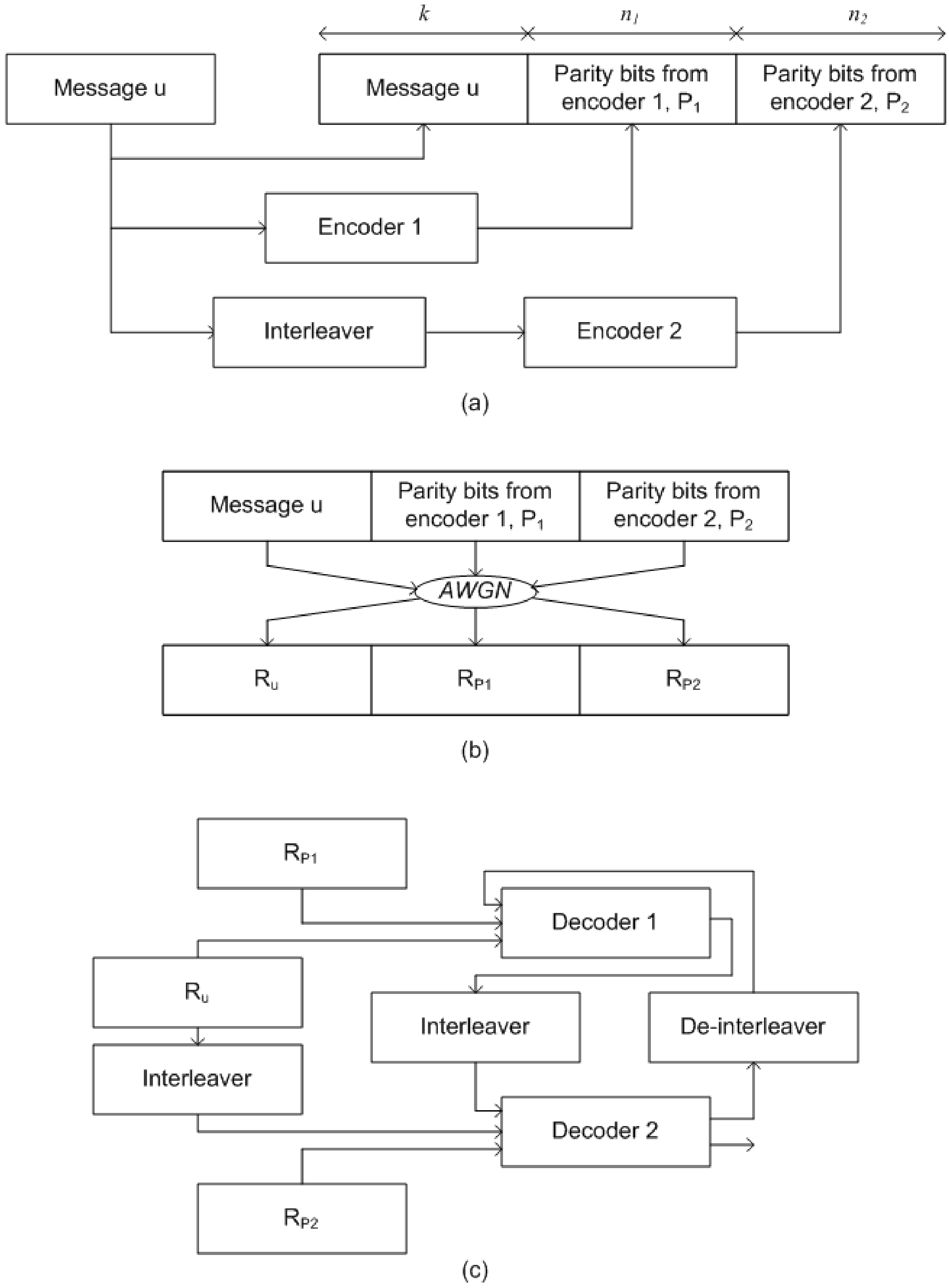

Figure 1: (a). The encoding process of turbo codes. (b) Additive White Gaussian Noise channel affecting the codeword. (c) The decoding process of turbo codes.

$$
\begin{matrix}
0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1
\end{matrix}
$$

Figure 2: A 12 by 16 parity-check matrix corresponding to a $(16, 4)$ LDPC Code

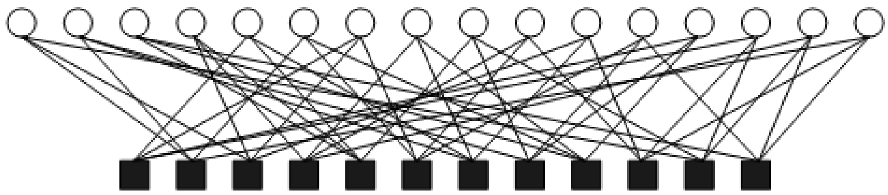

Figure 3: The bipartite graph of the $(16, 4)$ LDPC code with 12 check nodes and 16 bit nodes. Note that there is a one-to-one relationship between the edges of the bipartite graph and the non-zero entries of the parity-check matrix.

3. ENCODING AND DECODING OF TG CODES

The objective of devising TG codes (Soyjaudah & Catherine 2002) was to provide an appropriate architecture for concatenation of LDPC codes following a modified turbo style. Indeed, concatenating LDPC codes using the conventional method described in section 2.1 serves no purpose (MacKay 2000).

3.1 Encoding of TG Codes

The encoding is performed in three steps:

1. Generate two square matrices of size k by k. The message to be encoded is duplicated and placed in the two matrices.

2. An LDPC code Generator matrix G_1 of size n_1 by k is used for the encoding of each row of the first matrix (see Figure 4).

3. An LDPC code Generator matrix G_2 of size n_2 by k is used for the encoding of each column of the second matrix (see Figure 4).

The system may thus accommodate any overall rate $(k/n_1 + n_2) < \frac{1}{2}$ by adjusting the values of n_1, n_2 and k.

3.2 Decoding of TG Codes

The decoding is implemented in three steps: 1) the coarse decoding step, 2) post-decoding step, and 3) reconstruction step. We now describe each of these steps.

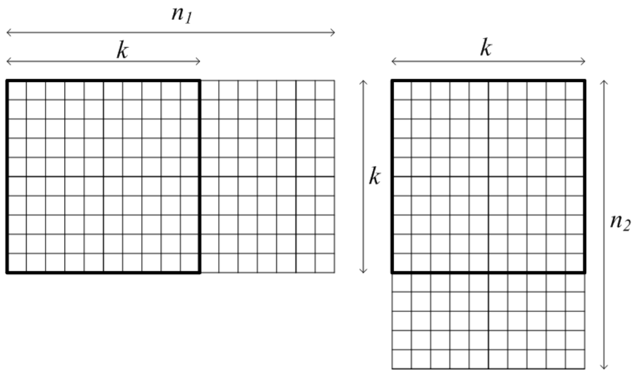

Figure 4: Encoding architecture of TG codes

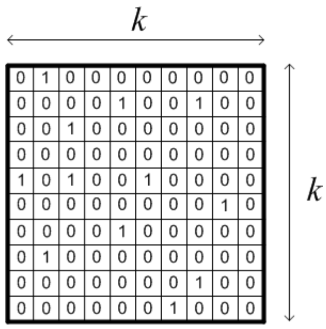

Figure 5: Example of a contradiction matrix. The non-zero entries represent positions at which the horizontal and vertical bits differ.

3.2.1 The coarse decoding step: We first proceed by building a *contradiction matrix*. This matrix captures all the positions in which the horizontal received vectors and vertical received vectors differ. An example of such a matrix is shown in Figure 5.

We next decode each row/column vector using the sum-product algorithm described in (MacKay 1999) in the order of increasing number of contradictions and store extrinsic information for use by vectors yet to be decoded. Using the example of Figure 5, row 4 and column 10 would be decoded first, whereas row 5 would be decoded last. Once the k horizontal decoders and the k vertical decoders have completed their task, we update the contradiction matrix and rerun the decoders. The first step stops when the contradiction matrix is an *all-zero* matrix, or when some maximum preset number of *super iterations* is reached.

3.2.2 The post-decoding step: Assuming the coarse decoding step have removed most of the errors from the two sets of vectors and that the contradiction matrix is *sparse* (but not all-zero), we activate this step. Strictly, we *erase* the bits corresponding to non-zero entries of the contradicion matrix (by setting their channel values to 0) and remove their associated extrinsic information. We then rerun the decoders for another preset number of super iterations. The "contradicted bits" being completely erased are more apt to receive information from their neighbor nodes and converge to their correct states. In practice the technique has been shown to be very effective (Catherine & Soyjaudah 2005).

3.2.3 The reconstruction step: Recall that two versions of the decoded bits are available: one from the horizontal set of decoders and one from the vertical set. This stage seeks to provide for the most efficient method of using both sets to reconstruct the k by k information block. In this section we shall demostrate two instances of the reconstruction process that depict the unique capabilites of TG codes.

1. ***Reconstruction in the presence of a block error.*** Let us assume that the ith horizontal vector from the horizontal set have not been decoded and that *all* the vertical vectors have been decoded correctly. In such case, the ith row of the k by k block is reconstructed using the ith bit from every vertical decoded vector. As such the system is fault tolerant and enables complete recovery in the presence of undecoded vectors provided however that one set is completely recovered.

2. ***Reconstruction in the presence of an "undetected error".*** Assume that *all* the vectors have been decoded and that the contradiction matrix is still non-zero as shown in Figure 6. Clearly, an "undetected error" is present. Using the example of Figure 6 a simple majority rule selects the 5th horizontal decoded vector as the faulty codeword. As such, instead of using the bit values of the 5th horizontal vector for reconstruction, the 5th bit of every vertical vector is used. A similar reasoning applies in case the "undetected eror is from the vertical set.

$$\overbrace{\begin{bmatrix} 0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0\\ 1&0&1&0&1&1&0&0&0&1\\ 0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0 \end{bmatrix}}^{k} \; k$$

Figure 6: Contradiction matrix used to recover from "undetected errors"

4. PERFORMANCE RESULTS AND ANALYSIS

We start this section with the decoding of a Portable Pixmap (PPM) image of 24 bit level, using both TG and LDPC codes. For this experiment, a 256 x 256 pixels PPM color lena image is used (see Figure 7). Note that half the number of super iterations set for TG codes is used for the coarse decoding step (see section 3.2.1). The other half is used for the post-decoding step (see section 3.2.2). From Figure 7 a difference of 3 orders of magnitude in terms of bit errors is noted. This good performance of TG codes may be attributed to its special faculties described in section 3.2.3. However the performance comes at at price: an increased complexity. Indeed the decoding of TG codes takes about 1 additional order of magnitude in amount of time taken when compared to LDPC codes. Nonetheless, the simulations assumes a serial implementation. In practice though, TG codes are easily amenable to *parallel* implementation, which decreases the decoding time drastically.

Next we consider the performance graph of the two codes. A close look at figure 8 reveals a coding gain of almost 3 dB in favor of the TG code at a Bit Error Rate (BER) of 10^{-5}. At this value of BER thus, the TG code consumes only half the power consumed by the conventional LDPC code.

Figure 7: Performance of a TG code with $k = 60$, $n_1 = n_2 = 120$ with vs. a conventional LDPC code with $k = 60$ and $n = 240$ when applied to the decoding of a 256 x 256 PPM lena image with 24 bits depth per pixel. Note that $E_b/N_o = 3$dB. The decoded images on the left column are LDPC decoded and those of the on the right column are TG decoded. Notice the large difference in bit errors of the two schemes.

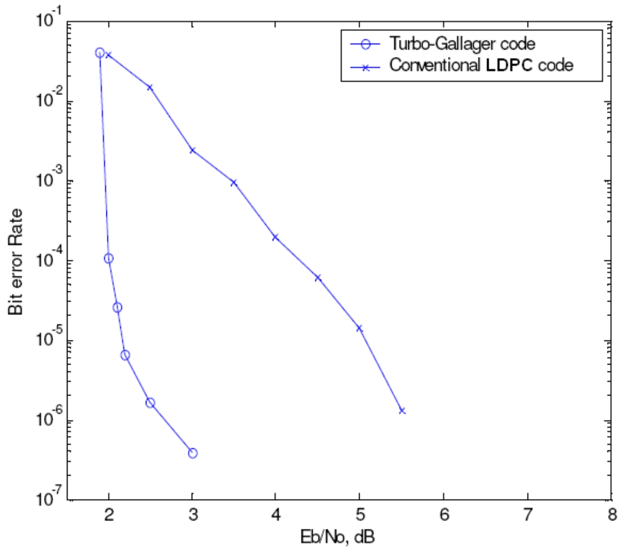

Figure 8: Bit error rate performance of a rate $\frac{1}{4}$ TG code with $k = 60$, $n_1 = n_2 = 120$ compared with a rate $\frac{1}{4}$ conventional LDPC code with $k = 60$ and $n = 240$.

5. CONCLUSION AND FUTURE WORKS

As demonstrated in this paper, TG codes provide a valuable alternative to turbo and LDPC codes for *power-efficient* applications. In this respect, deep space communication (where longer delays and higher complexity are acceptable) is an ideal system for the application of TG code. However additional research work is required to fully extract the power of the TG scheme. Specifically, a modification of the original TG code is required to support rates greater than half. Another important task is to devise a less complex decoding strategy that trades off performance for speed. This will bring TG codes to compete with the most powerful codes in a greater arena of applications. In any case however, it is hoped that the ideas behind TG codes will help in the development of future *intelligent* coding architectures (strongly related to neural networks) with capabilities exceeding those of the most powerful codes known to date.

Acknowledgment: The financial contribution of the Tertiary Education Commission is gratefully acknowledged.

References

Berrou, C., Glavieux, A. & Thitimajshima, P. (1993), Near Shannon limit error-correcting coding and decoding: Turbo codes, *in* ‘IEEE Int. Conf. on Communications (ICC-1993)’, pp. 1064–1070.

Burr, A. (2001), ‘Turbo-codes: the ultimate error control codes?’, *Electronics and Communication Engineering Journal* pp. 155–165.

Catherine, P. C. & Soyjaudah, K. M. S. (2005), Efficient decoding algorithm for turbo-gallager codes, *in* ‘Proceedings of the 8th International Symposium on DSP and Communication Systems’, Sunshine Coast, Australia, pp. 105–110.

CCSDS (2004), ‘CCSDS Gives SMART-1s Data Transmissions a Turbo Boost at the

Moon'. Press Release. Source: Consultative Committee for Space Data Systems (CCSDS).

CCSDS (2006), 'Experimental Specification CCSDS 131.1-O-1, Low-Density Parity-Check Codes For Use in Near-Earth and Deep Space Applications'.

ETSI (2000), 'Universal Mobile Telecommunications System (UMTS):Multiplexing and Channel Coding (FDD), 3GPP TS 125.212 version 3.4.0'.

ETSI (2005), 'Standard TR 102 376 V1.1.1: Digital Video Broadcasting (DVB) User guidelines for the second generation system for Broadcasting, Interactive Services, News Gathering and other broadband satellite applications (DVB-S2)'.

Gallager, R. (1963), *Low-Density Parity-Check Codes*, The M.I.T. Press, Cambridge, MA, USA.

Hagenauer, J. & Hoeher, P. (1989), A Viterbi algorithm with soft-decision outputs and its applications, *in* 'Proceedings IEEE Global Telecommunications Conference', IEEE, Dallas, TX, USA, pp. 1680–1686.

Kschischang, F., Frey, B. & Loeliger, H. (2001), 'Factor graphs and the sum-product algorithm', *IEEE Trans. Inf. Theory* **47**(2), 498–519.

L. Bahl, J. Cocke, F. J. & Raviv, J. (1974), 'Optimal decoding of linear codes for minimizing symbol error rate', *IEEE Trans. Inf. Theory* **20**(2), 284–287.

MacKay, D. (1999), 'Good error-correcting codes based on very sparse matrices', *IEEE Trans. Inf. Theory* **45**(2), 399–431.

MacKay, D. & Neal, R. (1995), Good codes based on very sparse matrices, *in* C. Boyd, ed., 'Cryptography and Coding. 5th IMA Conf., LNCS 1025', Springer, Berlin, pp. 100–111.

MacKay, D. J. C. (2000), Relationships between sparse graph codes, *in* 'Proceedings of the workshop on *Information-Based Induction Sciences (IBIS 2000), Shizuoka, Japan*', IEEE IT Japan Chapter, pp. 257–264.

Richardson, T. (2003), Error floors of LDPC codes, *in* 'Proc. of the 41st Annual. Allerton Conference on Communications, Control, and Computing', Monticello, IL, USA.

Shannon, C. (1948), 'A Mathematical Theory of Communication (continued)', *The Bell System Technical Journal* **27**(4), 623–656.

Soyjaudah, K. M. S. & Catherine, P. C. (2002), 'Performance of turbo-gallager codes over the additive white gaussian channel', *IEE Electronic letter* (20), 1187–1188.

Tanner, R. (1981), 'A recursive approach to low complexity codes', *IEEE Trans. Inf. Theory* **27**(5), 533–547.

Valenti, M. C. (1999), Iterative detection and decoding for wireless communications, PhD thesis, Virginia Polytechnic Institute and State University.

Valenti, M. C. & Woerner, B. D. (1997), Variable latency turbo codes for wireless multimedia applications, *in* 'Proc.Int. Symp. on turbo codes and related topics, Brest, France', pp. 216–219.