Irregular Repeat Accumulate Codes Based on Max-Flow Algorithm for Energy-Saving Networks

Ryuichi TATSUKAWA
The University of Electro-Communications
1-5-1 Chofugaoka, Chofu, Tokyo, JAPAN
tatsukawa@appnet.is.uec.ac.jp

Akiko MANADA
The University of Electro-Communications
1-5-1 Chofugaoka, Chofu, Tokyo, JAPAN
amanada@is.uec.ac.jp

Hiroyoshi MORITA
The University of Electro-Communications
1-5-1 Chofugaoka, Chofu, Tokyo, JAPAN
morita@is.uec.ac.jp

ABSTRACT

The efficiency of Error Correcting Codes (ECCs) arises when data should be transmitted via noisy channel. Many types of ECCs have been proposed so far, but ECCs with low encoding/decoding complexity are demanded when energy consumption is highly considered (*e.g.* networks using nanoscale devices). For such a situation, Irregular Repeat Accumulate (IRA) codes, a class of Low Density Parity Check (LDPC) codes, can be suitable candidates as ECCs since their encoding and decoding complexities are known to be low. In this paper, we introduce a construction method of IRA codes based on Max-flow algorithm and show the performance of robustness for the Binary Erasure Channel (BEC).

Categories and Subject Descriptors

E.4 [**Data**]: Coding and Information Theory; G.2.2 [**Mathematics of Computing**]: Graph Theory

General Terms

Theory

Keywords

Max-flow algorithm, Tanner graphs, IRA code, LDPC code

1. INTRODUCTION

The efficiency of Error Correcting Codes (ECCs) arises when data should be transmitted via noisy channel. A typical noisy channel would be Binary Erasure Channel (BEC) in which each bit $b \in \{0, 1\}$ is (independently) erased with erasure probability p (and b is sent without error with probability $1-p$). For example, in Wireless Body Area Networks (WBANs), data erasure due to communication jamming of human bodies etc. is a significant issue, and a suitable ECC would be useful to redeem erased data, in stead of asking retransmissions.

BODYNETS 2014, September 29-October 01, London, Great Britain

DOI 10.4108/icst.bodynets.2014.256920

Recently, Low Density Parity Check (LDPC) codes, proposed by Gallager in 1963 [1], have been gaining particular interest since they have shown to have high error-correcting performance (as the length of codewords goes to infinity). LDPC codes have sparse parity check matrices; that is, the number of 1's in parity check matrices is much smaller than the number of 0's. This property reduces the computation complexity of the Sum-product decoding, a decoding algorithm for LDPC codes (see, for example, [4]). The parity check matrices of LDPC codes can be represented by bipartite graphs called Tanner graphs consisting of the set of variable nodes and the set of check nodes. Indeed, there is a one-to-one correspondence between parity check matrices and Tanner graphs.

Amongst LDPC codes, we focus on Irregular Repeat Accumulate (IRA) codes [2], a class of LDPC codes, which have lower encoding complexity compared to general LDPC codes, in an attempt to use IRA codes as ECCs for energy-saving networks, such as WBANs. Some construction methods for IRA codes (irregular LDPC codes in general) have been proposed so far (*e.g.*, [5, 7]), but in this paper, we introduce a method based on Max-flow algorithm, an algorithm which outputs the maximum flow of a given network (see, for example, [6]). The method considers not only the degrees of variable nodes but also the degrees of check nodes when constructing Tanner graphs. In addition, it is simple and intuitive, and we basically do not have to care except for degrees of variable nodes and check nodes. We also present simulation results of IRA codes generated under the method to see the error-correcting performance for the BEC.

The rest of the paper is organized as follows. We explain fundamental background on LDPC codes and IRA codes, and their corresponding Tanner graphs in Section 2. In Section 3, we introduce our proposed algorithm, together with the review of Max-flow algorithm, and then in Section 4, we see the error-correcting performances of such IRA codes for the BEC. Finally, we will conclude the paper and address our future works in Section 5.

2. BASIC BACKGROUND

In this section, we go over fundamental notions which will be used in latter sections. In particular, we focus on LDPC codes and IRA codes, and their corresponding Tanner graphs.

2.1 LDPC codes and IRA codes

A *binary code* $\mathcal{C}$ is a set of finite-length vectors over $\mathbb{F}_2 = \{0, 1\}$, and we call an element $\boldsymbol{c}$ of $\mathcal{C}$ a *codeword*. We simply call $\mathcal{C}$ a code since we always assume $\mathcal{C}$ to be binary.

An *encoding* is a procedure to transform a message $\boldsymbol{m} = (m_1, m_2, ..., m_k) \in \mathbb{F}_2^k$ into a codeword $\boldsymbol{b} = (b_1, b_2, ..., b_n) \in \mathbb{F}_2^n$, where $k < n$, so that there is a one-to-one corresponding between messages and codewords. A linear code is a code $\mathcal{C}$ which can be represented as the kernel of the *parity check matrix* $\boldsymbol{H} = [h_{ij}]$, where $\boldsymbol{H}$ is a 0-1 matrix of size $m \times n$ with $m = n - k$. More precisely, $\mathcal{C}$ is expressed as

$$\mathcal{C} = \{\boldsymbol{b} \in \mathbb{F}_2^n : \boldsymbol{b}\boldsymbol{H}^T = \boldsymbol{0}\},$$

where $\boldsymbol{H}^T$ is the transpose of $\boldsymbol{H}$.

A *Low Density Parity Check (LDPC)* code is a linear code such that its parity check matrix $\boldsymbol{H}$ is a sparse matrix. An *Irregular Repeat Accumulate (IRA)* code is a LDPC code whose parity check matrix $\boldsymbol{H}$ is divided into two matrices $\boldsymbol{H}_u$ and $\boldsymbol{H}_p$ as

$$\boldsymbol{H} = [\boldsymbol{H}_u | \boldsymbol{H}_p],$$

where $\boldsymbol{H}_u$ is an $m \times k$ random sparse matrix and $\boldsymbol{H}_p$ is an $m \times m$ lower triangular matrix described below.

$$\boldsymbol{H}_p = \begin{pmatrix} 1 & 0 & 0 & 0 & ... & 0 & 0 \\ 1 & 1 & 0 & 0 & ... & 0 & 0 \\ 0 & 1 & 1 & 0 & ... & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & ... & 1 & 0 \\ 0 & 0 & 0 & 0 & ... & 1 & 1 \end{pmatrix}$$

An IRA code is a *systematic* code; that is, for a codeword $\boldsymbol{b} = (b_1, b_2, \ldots, b_n)$, the first k bits $(b_1, b_2, \ldots, b_k)$ correspond to the message $(m_1, m_2, \ldots, m_k)$ to be transmitted (the information bits), and the remaining $n - k(= m)$ bits $(b_{k+1}, b_{k+2}, \ldots, b_n)$ correspond to parity bits. We denote by $(x_1, x_2, \ldots, x_k)$ and $(y_1, y_2, \ldots, y_m)$ the information bits and the parity bits of a codeword, respectively.

An IRA code has an advantage that its encoding complexity is reasonably low. Indeed, for encoding, we only need to calculate parity bits $(y_1, y_2, \ldots, y_m)$, and each y_i is given by (setting $y_0 = 0$ by convention)

$$y_i = \sum_{j=1}^{k} h_{ij}^{(u)} x_i + y_{i-1} = \sum_{j \in supp(\mathbf{r}_i)} x_j + y_{i-1}, \qquad (1)$$

where $h_{ij}^{(u)}$ is the (i, j) component of $\boldsymbol{H}_u$, and $supp(\mathbf{r}_i)$ is the set of positions of 1's at the i-th row $\mathbf{r}_i$ of $\boldsymbol{H}_u$. Hence, IRA codes would be good candidate as ECCs for networks considering power consumption.

2.2 Tanner graphs

A graph $G = (V, E)$ consists of its vertex set V and edge set $E \subset V \times V$. We always assume that a graph is *simple*; that is, it has no multiple edges and no self-loops. For a vertex $v \in V$, the *degree* of v, denoted by $deg(v)$, is the number of edges attached to v. When a graph G is a directed graph, we denote the sets of incoming edges of v and outgoing edges of v by $In(v)$ and $Out(v)$, respectively.

For a graph $G = (V, E)$, a *subgraph* of G is a graph $H = (V', E')$ such that $V' \subset V$ and $E' \subset E$. In particular, H is called a *spanning subgraph* of G if $V' = V$.

Most of the graphs we focus on throughout this paper are *bipartite graphs*, where a graph $G = (V, E)$ is called bipartite if V can be partitioned into two disjoint sets (*partite sets*), V_1 and V_2, so that any pair of vertices in the same set are not adjacent. We denote a bipartite graph G by $G = (V_1 \dot{\cup} V_2, E)$, where $E \subset V_1 \times V_2$. A *complete bipartite graph* is a bipartite graph $G = (V_1 \dot{\cup} V_2, E)$ such that $E = V_1 \times V_2$.

A *Tanner graph* is a well known bipartite graph which is used to represent a parity check matrix. Indeed, we can consider one-to-one correspondence between a Tanner graph $G = (V_1 \dot{\cup} V_2, E)$ with $|V_1| = n, |V_2| = m$ and a (binary) $m \times n$ parity check matrix $\boldsymbol{H} = [h_{ij}]$, based on the condition that $h_{ij} = 1$ iff $(v_j, c_i) \in E$ for $v_j \in V_1$ and $c_i \in V_2$. Observe that $v_j \in V_1$ corresponds to the j-th bit b_j of a codeword, and each $c_i \in V_2$ is used to check whether the sum of adjacent vertices of c_i is 0 (over $\mathbb{F}_2$); that is, whether a condition to be a codeword is satisfied. Furthermore, it is straightforward to check that the weight of the i-th row (the sum of the entries in the i-th row) is equal to $deg(c_i)$, and the weight of the j-th column (the sum of the entries in the j-th column) is equal to $deg(v_j)$. Hereafter, we call vertices $v_1, v_2, \ldots, v_n$ in V_1 *variable nodes* and vertices $c_1, c_2, \ldots, c_m$ in V_2 *check nodes*, and use V and C to denote V_1 and V_2, respectively, to clarify the meaning of vertices.

Recall that a codeword of IRA codes is divided into the information bits $(x_1, x_2, ..., x_k)$ and the parity bits $(y_1, y_2, ..., y_m)$. So we further partition V into U and P so that the information bits correspond to vertices $u_1, u_2, ..., u_k$ in U (information nodes) and the parity bits correspond to vertices $p_1, p_2, ..., p_m$ in P (parity nodes). An example of a Tanner graph of an IRA code is given in Figure 1.

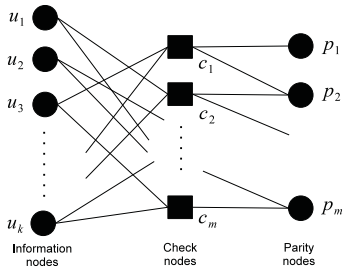


Figure 1: A Tanner graph of an IRA code

Using the Tanner graph, observe that (1) can be written as

$$y_i = \sum_{j \in N(i)} x_i + y_{i-1} \qquad (2)$$

where $N(i)$ is the index set of information nodes adjacent to check node c_i. We also note that for a parity check matrix $\boldsymbol{H} = [\boldsymbol{H}_u | \boldsymbol{H}_p]$ of an IRA code, $\boldsymbol{H}_u$ corresponds to the bipartite subgraph of the Tanner graph with U and C as

partite sets, and $\boldsymbol{H}_p$ corresponds to the bipartite subgraph with C and P as partite sets.

3. PROPOSED CONSTRUCTION METHOD FOR PARITY CHECK MATRICES

In this section, we explain our proposed method to construct parity check matrices of IRA codes. We first briefly recall the Max-Flow algorithm (also known as the Ford-Fulkerson algorithm), which outputs the maximum value of flows for a given directed graph. We then introduce how to construct parity check matrices based on the Max-Flow algorithm.

3.1 The Max-Flow algorithm

For a directed graph $\hat{G} = (\hat{V}, \hat{E})$, let us fix vertices s and t the source (starting vertex) and the sink (terminating vertex) in $\hat{G}$, respectively. A *network* $N = (\hat{G}, s, t, cap)$ is a quadruplet consisting of a directed graph $\hat{G}$, the source s, the sink t, and a function $cap : \hat{E} \to \mathbb{R}^+$, called the *capacity*, defined on $\hat{E}$. A *flow* over network N is also a function $f : \hat{E} \to \mathbb{R}^+$ defined on $\hat{E}$ satisfying the following two conditions:

Condition 1: The capacity constraint

For all $\hat{e} \in \hat{E}$, $f(\hat{e}) \leq cap(\hat{e})$.

Condition 2: The flow conservation

For all $v \in \hat{V} \setminus \{s, t\}$, $\sum_{e \in In(v)} f(e) = \sum_{e \in Out(v)} f(e)$.

The Max-flow algorithm is, roughly speaking, an algorithm which gives us the maximum value of flows from the source s to the sink t for a network N. More precisely, given a network N as an input, the algorithm outputs

$$M := \max_{\text{flow } f} \sum_{e \in Out(s)} f(e) = \max_{\text{flow } f} \sum_{e \in In(t)} f(e),$$

together with $f(\hat{e})$ for all $\hat{e} \in \hat{E}$. The value M is called the *maxflow* of $\hat{G}$. The details of the Max-flow algorithm can be found in any graph theory textbooks (see, for example, [6, Section 4.3]), so we omit them here due to limited margin.

3.2 Proposed construction method

We are now in a position of proposing our method to construct a parity check matrix $\boldsymbol{H} = [\boldsymbol{H}_u | \boldsymbol{H}_p]$ of an IRA code, based on the Max-flow algorithm. Recall that for parity check matrices of IRA codes, though $\boldsymbol{H}_p$ is fixed, $\boldsymbol{H}_u$ is unfixed. So we explain how to generate $\boldsymbol{H}_u$ as we desire.

Suppose that we want to generate an $m \times k$ random sparse matrix $\boldsymbol{H}_u$ such that the weight of the i-th row is $\rho_i (\leq k)$ and the weight of the j-th column is $\lambda_j (\leq m)$. Observe that it is equivalent to find a bipartite graph H with partite sets U and C such that

- $U = \{u_1, u_2, \ldots, u_k\}$ is the set of information nodes with $U = \{u_1, u_2, \ldots, u_k\}$ and $C = \{c_1, c_2, \ldots, c_m\}$ is the set of check nodes with $|C| = m$; and
- $deg(c_i) = \rho_i$ for $c_i \in C$ and $deg(u_j) = \lambda_j$ for $u_j \in U$.

The details of the algorithm are described in Algorithm 1. Key points of the algorithm are 1) H is a spanning subgraph of the complete graph $G = (U \dot{\cup} C, U \times C)$ with $U = \{u_1, u_2, \ldots, u_k\}$ and $C = \{c_1, c_2, \ldots, c_m\}$, and 2) it is possible to define the capacity $cap; \hat{E} \to \mathbb{R}^+$ as described since each $u_j \in U$ (*resp.*, $c_i \in C$) in G satisfies $deg(u_j) = m$ (*resp.*, $deg(c_i) = k$). We note here that the algorithm can be applied when we want to find a Tanner graph for LDPC codes, by changing U (the set of information nodes) to V (the set of variable nodes).

Algorithm 1 Proposed algorithm

Require: the complete bipartite graph $G = (U \dot{\cup} C, U \times C)$ with $U = \{u_1, u_2, \ldots, u_k\}$ and $C = \{c_1, c_2, \ldots, c_m\}$.

1: add the source s and the sink t to G, where s is located on the left side of U and t is on the right side of C.
2: assign directed edges from s to $u_j \in U$ and those from $c_i \in C$ to t, and name the resulting graph $\hat{G} = (\hat{V}, \hat{E})$.
3: define $cap : \hat{E} \to \mathbb{R}^+$ for $\hat{e} \in \hat{E}$ so that

$$cap(\hat{e}) = \begin{cases} 1 & (\hat{e} \in U \times C) \\ \lambda_j & (\hat{e} = (s, u_j)) \\ \rho_i & (\hat{e} = (c_i, t)), \end{cases}$$

to generate $N = (\hat{G}, s, t, cap)$.
4: use the Max-Flow algorithm for N and compute the maxflow M.
5: **if** $M = \sum_{j=1}^{k} \lambda_j (= \sum_{i=1}^{m} \rho_i)$, **then**
6: delete edges e with $f(e) = 0$, and s, t together with their attached edges from $\hat{G}$.
7: **return** the resulting graph H.
8: **else**
9: **return** "There is no such a bipartite graph."
10: **end if**

Using the proposed algorithm, we can randomly construct parity check matrices which have not only desired variable node degrees but also desired check node degrees. In addition, the algorithm runs in polynomial time, is simple and easy to follow. Indeed, we believe that our construction method will be helpful to find better IRA codes.

4. SIMULATION RESULTS

In this section, we present simulation results of IRA codes constructed under our proposed algorithm. In this simulation, we convert a length-k random message into a length-n codeword of an IRA code, where the codeword length n is set to be reasonably short (256, 512 or 1024) with an aim to apply IRA codes to energy-saving networks. The code rate k/n is always 1/2. We examine the error-correcting performance of IRA codes using the Bit Error Rate (BER) given by

$$\text{BER} = \frac{\text{the number of incorrect bits after decoding}}{\text{the codeword length } n \text{ (the total number of bits)}},$$

and the number of success (the number of trials at which all bits are correctly decoded) amongst 1,000 trials. Throughout the simulations, we set the fractions Φ_i's of information nodes of degree i to $\Phi_3 = 0.543, \Phi_4 = 0.102, \Phi_5 = 0.008, \Phi_6 = 0.020, \Phi_7 = 0.008, \Phi_8 = 0.008, \Phi_9 = 0.047$ and $\Phi_{10} = 0.266$, in accordance with [5]. The graphs for BER are always semi-logarithmic and the points of BER= 0 (under the simulations) are not plotted.

We first check error-correcting performances with respect to (w.r.t.) codeword length n (Figure 2), where the degrees of check nodes is set to be equal. We can see some significant improvements as n gets large. To maintain reasonable performance, we think that n should be at least 512.

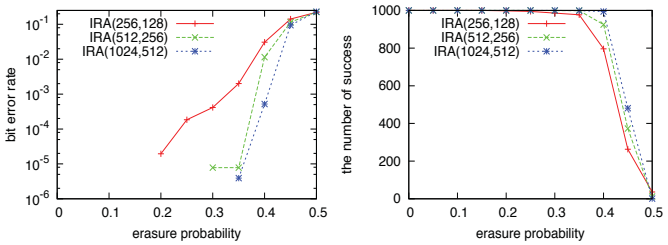


Figure 2: The comparison of error-correcting performance w.r.t. codeword lengths

We next compare the following four types when the codeword length $n = 512$ or 1024.

Type 1 (Red line) : We first set the degrees of check nodes to be equal (as much as possible); that is, the fractions Ω_j's of check nodes of degree j are $\Omega_5 = 0.61$ and $\Omega_6 = 0.39$.
Type 2 (Green line): We slightly changed the the degree of check nodes based on Type 1. The fractions Ω_j's of check nodes of degree j are $\Omega_4 = 0.10$, $\Omega_5 = 0.51$, $\Omega_6 = 0.29$ and $\Omega_7 = 0.10$.
Type 3 (Blue line): We further changed the degree of check nodes based on Type 2. The fractions Ω_j's of check nodes of degree j are $\Omega_4 = 0.20$, $\Omega_5 = 0.41$, $\Omega_6 = 0.19$ and $\Omega_7 = 0.20$.
Type 4 (Pink line): We theoretically investigated a candidate of good degrees of check nodes based on density evolution of LDPC codes (recall that IRA codes are in the class of LDPC codes); see, for example, [3, Section 7.2]. The fractions Ω_j's of check nodes of degree j are $\Omega_4 = 0.31$, $\Omega_6 = 0.06$ and $\Omega_7 = 0.63$.

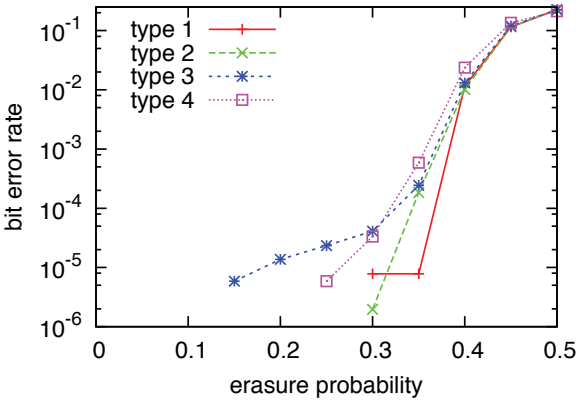


Figure 3: The comparison of BER for $(n, k) = (512, 256)$.

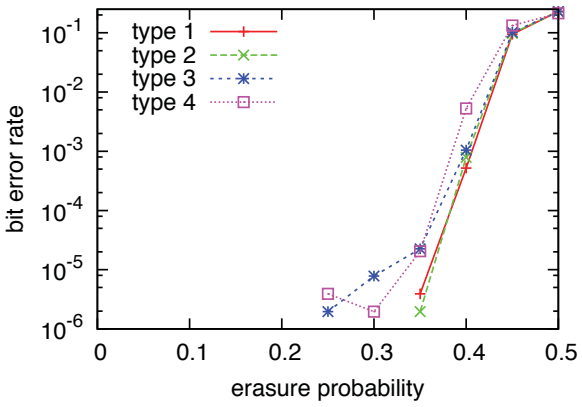
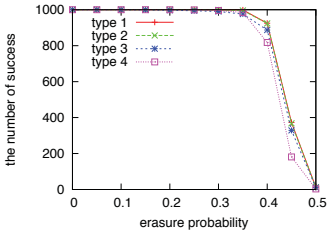
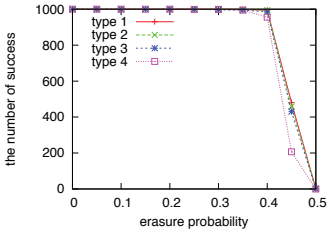


Figure 4: The comparison of BER for $(n, k) = (1024, 512)$.



era. prob. (p)	type 1	type 2	type 3	type 4
0.05	1000	1000	1000	1000
0.10	1000	1000	1000	1000
0.15	1000	1000	999	1000
0.20	1000	1000	997	1000
0.25	1000	1000	994	999
0.30	999	999	990	996
0.35	999	996	976	985
0.40	925	924	887	818
0.45	373	368	329	181
0.50	12	12	11	3

Figure 5: The comparison of the number of success for $(n, k) = (512, 256)$. The table shows the exact numerical results.



era. prob. (p)	type 1	type 2	type 3	type 4
0.05	1000	1000	1000	1000
0.10	1000	1000	1000	1000
0.15	1000	1000	1000	1000
0.20	1000	1000	1000	1000
0.25	1000	1000	999	999
0.30	1000	1000	997	999
0.35	999	999	994	995
0.40	994	994	986	956
0.45	481	459	430	207
0.50	1	3	0	0

Figure 6: The comparison of the number of success for $(n, k) = (1024, 512)$. The table shows the exact numerical results.

For the BEC, the Shannon capacity (the maximum code rate for which messages can be reliably transmitted) is known to be $1 - p$. Our simulation shows that each type has good error-correcting performance, and in particular, Type 1 (an IRA code with equal check node degree) gives the best performance. On the other hand, the theoretical results show that Type 4 is the best amongst the 4 types. We think that the gap between theoretical results and simulation results comes from the fact that the theoretical results are based upon the non-existence of cycles in a Tanner graph (as the codeword length n goes to infinity). Hence, we hope to modify our method so that we can increase the girth (the length of a shortest cycle) in a Tanner graph to construct better IRA codes.

5. CONCLUSION

In this paper, we proposed an algorithm using the Max-Flow algorithm to construct parity check matrices of IRA codes. The algorithm is simple and intuitive, and can consider both the degrees of variable nodes and the those of check nodes. We also presented the simulation results of IRA codes constructed under the algorithm for the BEC.

As a future work, we aim to find good IRA codes with shorter codeword length for a practical use. On the other hand, for IRA codes with short codeword length, short cycles in their Tanner graphs can decrease the decoding performance. Hence, we will also attempt to modify our method to remove such short cycles.

6. ACKNOWLEDGEMENTS

This work was supported by JSPS KAKENHI Grant Number 25870228.

7. REFERENCES

[1] R. G. Gallager,"Low-Density Parity-Check Codes," in Research Monogragh series., Cambridge, MA:MIT Press, 1963.

[2] H. Jin, A. Khandekar and R. McEliece, "Irregular repeat-accumulate codes," *Proc. 2nd Int. Symp. Turbo Codes Related Topics*, pp. 1-8, Sept. 2000.

[3] S. J. Johnson, "Iterative Error Correction," Cambridge University Press, 2010.

[4] F. R. Kschischang, B. J. Frey and H-A. Loeliger, "Factor Graphs and the Sum-product Algorithm," *IEEE Trans. Inf. Theory*, vol.47, no.2, pp. 498-519, Feb. 2001.

[5] G. Liva, P. Pulini and M. Chiani, "On-Line Construction of Irregular Repeat Accumulate Codes for Packet Erasure Channels," *IEEE Trans. Wireless Commun.*, vol. 12, no.2, pp. 680-689, Feb. 2013.

[6] D. B. West, "Introduction to Graph Theory (second edition)," Prentice Hall, 2000.

[7] J. Xu, L. Chen, I. Djurdjevic, S. Lin and K. Abdel-Ghaffar, "Construction of Regular and Irregular LDPC Codes: Geometry Decomposition and Masking," *IEEE Trans. Inf. Theory*, vol.53, no.1, pp. 121-134, Jan. 2007.