Scheduled-PEG construction of LDPC codes for Upper-Layer FEC

Lam Pham Sy, Valentin Savin, David Declercq, Nghia Pham

Scheduled-PEG construction of LDPC codes for Upper-Layer FEC

Lam Pham Sy[1], Valentin Savin[2], David Declercq[3], and Nghia Pham[1]

[1] Eutelsat, Paris, France
lphamsy@eutelsat.fr, HPham@eutelsat.fr
[2] CEA-LETI, MINATEC campus, Grenoble, France
valentin.savin@cea.fr
[3] ENSEA, Cergy-Pontoise, France
declercq@ensea.fr

Abstract The Progressive Edge Growth (PEG) algorithm is one of the most widely-used methods for constructing finite length LDPC codes. In this paper we consider the PEG algorithm together with a *scheduling distribution*, which specifies the order in which edges are established in the graph. The goal is to find a scheduling distribution that yields "the best" performance in terms of *decoding overhead*, performance metric specific to erasure codes and widely used for upper-layer forward error correction (UL-FEC). We rigorously formulate this optimization problem, and we show that it can be addressed by using genetic optimization algorithms. We also exhibit PEG codes with optimized scheduling distribution, whose decoding overhead is less than half of the decoding overhead of their classical-PEG counterparts.

Keywords: LDPC codes, bipartite graph, PEG, UL-FEC, decoding overhead/inefficiency.

1 Context and motivations

Low Density Parity Check (LDPC) codes constitute a very broad class of FEC codes, distinguished by the fact that they are defined by sparse parity-check matrices, and can be iteratively decoded in linear time with respect to their block-length. Invented by Gallager in early 60's [1], but considered impractical to implement, these codes have been neglected for more that three decades, and "rediscovered" in the late 90's [2]. Nowadays, a large body of knowledge has been acquired (analysis, optimization, construction); LDPC codes are known to be capacity approaching codes for a large class of channels [3], and became synonymous with modern coding.

However, this capacity approaching property holds in the asymptotic limit of the code length, and codes optimized from this asymptotic perspective may suffer significant performance degradation at practical lengths. Actually, the asymptotic optimization, performed by using density-evolution methods [4], yields an

irregularity profile, which specifies the distribution of node-degrees in the bipartite (Tanner) graph [5] associated with the code. It is assumed that the girth[4] of the bipartite graph goes to infinity with the code-length. Hence, optimized irregularity profiles can be used to construct codes that are "long enough" (at least few thousand bits) to avoid short cycles, although they must be "short enough" to be practical.

One of the most widely-used method for constructing finite length codes is the Progressive Edge Growth (PEG) algorithm [6]. It constructs bipartite graphs with large girth, by establishing edges progressively: the graph grows in an edge-by-edge manner, optimizing each local girth. There is an *underlying edge order* within the PEG, corresponding to the order in which edges are established in the graph. In general, edges are progressively established starting with those incident to symbol-nodes of degree-2 and ending with those incident to symbol-nodes of maximum degree. However, any other order with respect to the symbol-node degrees would also be possible. Besides, for a given symbol-node degree, edges can be established in a *node-by-node* manner (all edges incident to some symbol node are established before moving to the next symbol-node), or in a *degree-by-degree* manner (a first edge is established for each symbol-node, then a second edge is established for each symbol-node, and so on until all the symbol-nodes reach the given degree). Although this order may significantly impact the performance of the constructed code, it is rather difficult to formalize and has practically not been investigated in the literature. There are however several papers that aim to enhance the PEG construction by optimizing some objective function, as for instance minimizing the number of cycles created [7], or minimizing the approximate cycle extrinsic (ACE) message degree [8], [9].

2 Scheduled-PEG construction

In this paper we consider the PEG algorithm together with a *scheduling distribution*, which will be referred to as scheduled-PEG, or SPEG for short. Within the SPEG algorithm, symbol-nodes are divided into subsets, each subset containing symbol-nodes of same degree. Edges incident to the symbol-nodes of a subset are established in a degree-by-degree manner, before moving to the next subset. The scheduling distribution specifies the fraction of nodes within each subset. Our purpose is to find a scheduling distribution that yields the best performance in terms of decoding overhead (performance metric widely used for UL-FEC). We rigorously formulate this optimization problem, and we show that it can be addressed by using genetic optimization algorithms.

The proposed Scheduled-PEG algorithm allows the enhancement of the classical PEG algorithm. By varying the scheduling distribution, one can explore the ensemble of LDPC codes with fixed code-length and degree distributions, which allows us to find codes with very small average inefficiency (the performance metric considered in this paper).

[4] Length of a shortest cycle.

We show that the SPEG algorithm can be successfully combined with genetic optimization algorithms[10], which significantly improves the average inefficiency of the constructed LDPC codes over the classical-PEG construction. In terms of error rate curves, this translates into a significant improvement of the waterfall region.

Finally, we remark that the optimization of the scheduling distribution makes use of the specific channel model (through the use of the decoding inefficiency). Hence, LDPC codes constructed by using the SPEG algorithm together with an optimized scheduling distribution are channel dependent. However, the proposed algorithm could be generalized for more general channel models (e.g. by optimizing with respect to a target FER, or to the area under the FER curve.).

Acknowledgment

This work was supported by the French National Research Agency (ANR), grant No 2009 VERS 019 04 – ARSSO project.

The complete version of the paper can be found on hal.inria.fr and arxiv.org websites.

References

1. R. G. Gallager, *Low Density Parity Check Codes*, M.I.T. Press, 1963, Monograph.
2. D. J. C. MacKay, "Good error-correcting codes based on very sparse matrices," *IEEE Trans. Inform. Theory*, vol. 45, no. 2, pp. 399–431, 1999.
3. T. J. Richardson, M. A. Shokrollahi, and R. L. Urbanke, "Design of capacity approaching irregular low density parity check codes," *IEEE Trans. Inform. Theory*, vol. 47, no. 2, pp. 619–637, 2001.
4. T. J. Richardson and R. L. Urbanke, "The capacity of low-density parity-check codes under message-passing decoding," *IEEE Trans. Inform. Theory*, vol. 47, no. 2, pp. 599–618, 2001.
5. R. M. Tanner, "A recursive approach to low complexity codes," *IEEE Transactions on Information Theory*, vol. 27, no. 5, pp. 533–547, 1981.
6. X. Y. Hu, E. Eleftheriou, and D. M. Arnold, "Regular and irregular progressive edge-growth tanner graphs," *IEEE Trans. Inform. Theory*, vol. 51, no. 1, pp. 386–398, 2005.
7. A. Venkiah, D. Declercq, and C. Poulliat, "Design of Cages with a Randomized Progressive Edge Growth Algorithm," *IEEE Commun. Letters*, vol. 12, no. 4, pp. 301–303, 2008.
8. T. Tian, C. Jones, J. D. Villasenor, and R. D. Wesel, "Construction of irregular LDPC codes with low error floors," *IEEE Trans. Commun.*, vol. 52, pp. 12421247, 2004.
9. H. Xiao and A. H. Banihashemi, "Improved progressive-edge-growth (PEG) construction of irregular LDPC codes," *IEEE Commun. Letters*, vol. 8, no. 12, pp. 715–717, 2004.
10. R. Storn and K. Price, "Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces," *J. of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.