On the Design of LDPC-Convolutional Ensembles Using the TEP Decoder

Pablo M. Olmos, *Member, IEEE*, Luis Salamanca, *Student Member, IEEE*, Juan José Murillo-Fuentes, *Senior Member, IEEE*, and Fernando Pérez-Cruz, *Senior Member, IEEE*

Abstract—Low-density parity-check convolutional (LDPCC) codes asymptotically achieve channel capacity under belief propagation (BP) decoding. In this paper, we decode LDPCC codes using the Tree-Expectation Propagation (TEP) decoder, recently proposed as an alternative decoding method to the BP algorithm for the binary erasure channel (BEC). We show that, for LDPCC codes, the TEP decoder improves the BP solution with a comparable complexity or, alternatively, it allows using shorter codes to achieve similar error rates. We also propose a window-sliding scheme for the TEP decoder to reduce the decoding latency.

Index Terms—convolutional LDPC codes, finite-length analysis, tree-structured expectation propagation.

I. INTRODUCTION

Low-density parity-check convolutional (LDPCC) ensembles were first proposed in [1] to construct capacity achieving (c.a.) codes under belief propagation (BP) decoding [2]. The construction of LDPCC ensembles can be cast as the concatenation of standard LDPC codes with adequate termination constraints [3], [4]. Unlike irregular c.a. LDPC ensembles [2], LDPCC codes are known to be asymptotically good under BP decoding, in the sense they achieve channel capacity with a minimum distance that linearly grows with the constraint length [5], [6].

LDPCC codes are based on L parallel regular LDPC codes with M variables nodes each, in which nearby codes interchange check nodes [3]. The LDPC codes at the extremes are terminated incorporating new check nodes, which are intimately related with the good asymptotic properties of LDPCC codes, but produce an undesirable rate loss, proportional to $1/L$ in the finite-length regime [4], [5]. As a consequence, long chains of LDPC codes (i.e. L) are needed to mitigate this rate loss, yielding codes with excessive code lengths and decoding latency, i.e. a delay associated to the acquisition and decoding of a new received word [7]. Besides, M has to grow at a faster rate than L for LDPCC codes to approach capacity [4].

The tree-structured expectation propagation (TEP) decoder [8] improves the BP algorithm solution, by using pairwise

marginal estimates to improve each bit posterior probability estimate. As a consequence, when used for LDPC decoding, it is able to break some fairly common *stopping-sets* (SSs) that limit the BP algorithm [9], improving the decoding performance. Besides, for the binary erasure channel (BEC), the TEP complexity is linear with the code length, as in BP decoding. For finite-length LDPC ensembles, we show in [9] that the TEP either provides a gain in word error rate (WER) for a given code length, or in a reduction of the code size needed to achieve a target WER. However, the gain provided by the TEP when compared to BP vanishes with the block length for LDPC ensembles.

In this letter, we advance the TEP decoding for LDPCC codes over the BEC [8]. We focus our study on the reduced WER provided by the TEP decoder compared to BP decoding, or similarly on the reduction in the LDPCC code length for a desired WER. The TEP decoder improves BP for LDPCC codes of interest, because for low rate-loss finite codes, it implies a low M , which is the critical value for the TEP decoder to improve BP. So, even for extremely large codes, we show that the TEP decoder exhibits a significant gain with respect to BP, making the TEP decoder an attractive tool to exploit LDPCC codes in erasure coding applications [10], [11], [12]. To address the decoding latency, windowed BP (wBP) has been proposed to provide a simple, yet efficient, way to trade-off decoding performance and reduced latency [3], [13]. In this letter, we also put forward a windowed TEP (wTEP) decoder.

II. BP AND TEP DECODING OF BLOCK CODES OVER BEC

The TEP decoder for the erasure channel is described in [8], [9] as a peeling-type algorithm [2], which works over the Tanner graph of the code. To describe the TEP algorithm we assume, without loss of generality, that the all zero codeword is transmitted through a BEC¹ with erasure parameter ϵ . The Tanner graph is initialized by removing all the non-erased variables, along with all their connections. Then, one of the two following steps is performed at each iteration:

- 1) **Variable identification:** Look for a degree-one check node in the graph and remove it along with the variable it is connected to. The parity of the check node indicates the value of the variable, which is declared as revealed.
- 2) **Graph reduction:** Look for a degree-two check node in the graph and remove it along with one of the variables it is connected to. The parity of the check node indicates if

¹The full algorithm is described in [8].

This work was partially funded by Spanish government (Ministerio de Educación y Ciencia, TEC2009-14504-C02-01,02, Consolider-Ingenio 2010 CSD2008-00010), Universidad Carlos III (CCG10-UC3M/TIC-5304) and European Union (FEDER).

Pablo M. Olmos and F. Pérez-Cruz are with Dept. Teoría de la Señal y Comunicaciones, Universidad Carlos III de Madrid (Spain). E-mail: {olmos, fernando}@tsc.uc3m.es

Luis Salamanca and J.J. Murillo-Fuentes are with the Dept. Teoría de la Señal y Comunicaciones, Escuela Técnica Superior de Ingeniería, Universidad de Sevilla, Spain. E-mail: {salamanca, murillo}@us.es

Fig. 1. In (a), V_1 and V_2 , that share a check node of degree two. In (b), V_1 heirs the connections of V_2 (solid lines) and P_1 and V_2 are removed.

the variables are equal or opposite. Hence, the remaining variable can represent the removed variable in the graph, i.e. it inherits its connections. Note that the removed variable will be revealed once the remaining variable in the graph is revealed. This process is sketched in Fig. 1. The variable V_1 heirs the connections of V_2 (solid lines) in Fig. 1(b) and the check P_1 and the variable V_2 are removed.

The BP decoder as a peeling-type algorithm only implements the first step, i.e. variable identification. Hence, the TEP can be seen as a procedure to improve BP once it is unable to decode. The TEP operates as described until all variable nodes have been removed, i.e. successful decoding, or until there are no degree-one and degree-two check nodes left, i.e. unsuccessful decoding. In [9], it is shown that the TEP decoder eventually creates degree-one check nodes, breaking some of the SSs that degrade the BP solution [2]. The TEP decoder complexity is analyzed in [8], where we show that the removal of a degree-two check node is performed by a basic BP iteration followed by the addition of two columns of the parity check matrix. Along the decoding process, the matrix remains sparse and the resulting complexity is of order $\mathcal{O}(n)$.

III. CONVOLUTIONAL LDPC ENSEMBLES

The construction of LDPCC codes used in this letter are based on the concatenation of regular (l, r) LDPC codes, with l ones per column, r ones per row and $k = r/l \in \mathbb{N}$. More flexible structures and constructions can be found in [3], [4], which are amenable to TEP decoding as well, but which cloud the presentation of this letter. To build an LDPCC code, we first take a set of L regular (l, r) LDPC codes with M variable nodes and M/k check nodes. We label each code from 1 to L and we refer to them as sections. Now we construct the convolutional code by reconnecting at random the u -th edge of every variable node at section i to a check node in section $i + u - 1$, for $u = 2, \dots, l$. So any variable node in section i ends up being connected to a check node in section $i, i + 1, \dots, i + l - 1$ for $i = 1, \dots, L$. To terminate the code we need to include $l - 1$ additional sections constituted only by M/k check nodes for the variables from sections $L - l + 1$ to L to be connected to. These additional constraints explain the rate loss produced when constructing LDPCC codes. The convolutional code obtained in this way is referred to as the (l, r, L, M) LDPC convolutional ensemble [4].

Alternatively, we can describe the LDPCC code using its associate protograph, sometimes referred to as the *coupled or*

convolutional protograph, which serves as a layout to generate the code and it constitutes a useful tool to visualize and study the properties of the code [3], [5], [14]. We have depicted an example for an LDPCC coupled protograph in Fig. 2 (a) for $l = 3$, $r = 6$, and $L = 8$. Each check node represents a section, we have twice as many variables as check nodes, and each variable is connected to 3 sections.

The parity check matrix of the LDPCC has $n = ML$ variable nodes and $\frac{M}{k}(L + l - 1)$ check nodes, which yields a design rate r :

$$r(l, r = kl, L) = \frac{k - 1}{k} - \frac{(l - 1)}{kL}, \quad (1)$$

where the first term is the rate of the (l, r) -regular LDPC code [2] and the second, denoted by Δ_r , is a rate loss, caused by the inclusion of additional check nodes to terminated the code.

The asymptotic analysis of the LDPCC ensembles [3], [4] shows that:

$$\lim_{L \rightarrow \infty} \lim_{M \rightarrow \infty} P_b(l, r, L, M, \epsilon) = 0, \quad \epsilon \leq \epsilon_{\text{MAP}}(l, r), \quad (2)$$

where $P_b(l, r, L, M, \epsilon)$ is the ensemble average bit error rate, $\epsilon_{\text{MAP}}(l, r)$ is the maximum a posteriori (MAP) solution of the (l, r) -regular LDPC ensemble, and the limits are not exchangeable. Furthermore, $\epsilon_{\text{MAP}}(l, r)$ quickly tends to the channel capacity with l [2].

A study of the finite-length regime and the scaling behavior is provided in [5]. Despite no analytic behavior is provided, it is shown that, as we increase the ratio L/M , we degrade the decoding performance in both the waterfall and the error floor region. Therefore M should be significantly larger than L and L should be large enough to avoid great rate losses. For fixed codeword length $n = ML$, there is a trade-off between the minimization of the rate loss in (1) and the optimization of the code performance.

Fig. 2. In (a), coupled protograph associated to the LDPCC with $(l, r) = (3, 6)$ and $L = 8$. In (b), subgraph S_1 formed to perform windowed decoding with $W = 3$. The targeted variables are those filled in black.

A. Windowed BP and TEP schemes for reduced latency

As described in Section II, the complexity of the BP and the TEP decoders for the BEC is of order $\mathcal{O}(n)$ and, hence, we assume that any limitation in the code length n is due to decoding latency constraints. Windowed schemes were proposed to reduce the LDPCC latency for the BP decoder [3], [13], without significantly degrading the code performance.

The LDPC convolutional structure imposes a constraint on the variable nodes connections because a variable at section $i \in [1, L]$ is only connected to check nodes at sections

$i, \dots, i + l - 1$. This effect can be exploited to perform continuous decoding of the received stream through an sliding window along the bit sequence. Given the (l, r, L, M) parity check matrix and a window-size W , the windowed-BP (wBP) decoder operates as follows [13]. For $j = 1, \dots, L$,

- 1) Construct the subgraph $\mathcal{S}_j$ consisting on all check nodes corresponding to sections $[j, j + W]$ and all the variable nodes involved. For instance, in Fig. 2 (b), we depict the part of the coupled protograph that corresponds to subgraph $\mathcal{S}_1$ for the case $l = 3, r = 6, L = 8$ and $W = 3$.
- 2) Run the BP decoder within this windowed subgraph until the *targeted variables* (t.v.) are recovered. The t.v. are the set of variables corresponding to the first section of the subgraph $\mathcal{S}_j$. In Fig. 2 (b), we represent the t.v. by black nodes.
- 3) If the t.v. set for $\mathcal{S}_j$ has not been recovered, the wBP decoder halts.

This windowed scheme is easily extended to the TEP decoder by also considering the graph reduction step, Step 2) of the TEP algorithm. We refer to this configuration as the wTEP decoder. Note that the window size, W , ranges between $[l, L + l - 1]$ and that the solution is strictly suboptimal with respect to the BP/TEP algorithms for $W < L + l - 1$.

IV. LDPC DESIGN USING THE TEP DECODER

In the previous section we have described the construction of the (l, r, L, M) ensembles, pointing out their advantages as well as their limitations. As stated before, we require large L values to neglect the rate loss in (1), but the ratio L/M must tend to zero to improve the finite-length performance [5]. As a consequence, the code length $n = ML$ needs to be in general very large. In this section, we illustrate the benefits of using the TEP decoder compared to the BP decoder for finite length codes. For simplicity, we focus on the $(l = 3, r = 6, L, M)$ LDPC ensemble, which is the case for which the rate loss in (1) is smallest.

A. Gain in word error rate

We first compare both algorithms when no windowed scheme is considered. We assume a rate loss of $\Delta_r = 5\%$, from which the chain length L is calculated. In Fig. 3 we depict the word error rate (WER) for the $(3, 6, 20, M)$ LDPC ensemble, under BP (dashed lines) and TEP (solid lines) decoding, and $M = \{256, 512, 1024\}$. $L = 20$ is the minimum value that satisfies $\Delta_r \leq 5\%$. The results are averaged throughout 20 code samples and 10^4 channel realizations. In all cases the TEP significantly reduces the BP error rate.

For standard LDPC ensembles, we observed in [9] for regular and irregular c.a. ensembles that the TEP and BP converge in practice for code lengths over 10000 bits. On the contrary, for LDPC ensembles this is not the case and the TEP decoder significantly outperforms BP for much longer codes, because M , which plays the role of the code length for standard LDPC codes, is not sufficiently large.

B. Complexity

As discussed in Section II, the complexity of the graph reduction step in the TEP procedure is given by the sparsity of the parity matrix along decoding. To illustrate that the graph remains sparse and the TEP complexity for LDPC codes is of order $\mathcal{O}(n)$, in Fig. 4 we include the evolution of the mean variable degree, Λ_{avg} , averaged over 10^3 cases where the BP decoder gets stuck. We include a set of 20 decoding realizations chosen at random. The code was a $(3, 6, 20, 1024)$ LDPC and $\epsilon = 0.48$. As we can observe, the gain in performance reported in Fig. 3 for this case (curves with $\diamond$ marker) is achieved by processing degree-two check nodes at an average cost of no more than 12 binary sums.

C. Code length reduction

Since there is not significant increase in complexity, we can exploit the TEP decoder gain in performance to reduce the code-length needed to achieve a specific WER. For instance, note that the TEP error rates for $M = 256$ and $M = 512$ in Fig. 3 are, respectively, close to the BP error rates for 512 and 1024. Hence, there exists M values slightly above 256 and 512 for which the TEP performs as the BP with a code with approximately 50% less bits.

D. Latency reduction

In practice, the minimization of the rate loss in (1) is of extreme importance, and usually large L values are needed. For instance, in the $(3, 6)$ case, a rate loss $\Delta_r = 0.05\%$ requires a chain length of $L = 100$. Such long chains may require the use of the windowed decoder scheme, described in Section III-A, to reduce the decoding latency. In Table I and Table II we respectively summarize the WER computed for the wBP and the wTEP decoders for the $(3, 6, 100, 512)$ ensemble at different ϵ values. Like in the unwinded case in Fig. 3, the wTEP decoder noticeable improves the wBP performance even though the code length in this scenario is extremely large, around 50000 bits. In this case, the wTEP gain in performance can be exploited not only to gain either a WER

Fig. 3. TEP (solid) and BP (dashed) performance for the $(3, 6, 20, M)$ ensemble with $M = 256$ ($\circ$), $M = 512$ ($\square$) and $M = 1024$ ($\diamond$).

Fig. 4. Evolution of the variable node degree Λ_{avg} as the TEP decodes the $(3, 6, 20, 1024)$ ensemble for $\epsilon = 0.48$, average (solid) and individual decoding trajectories (dashed).

or a code length reduction, but to tighten the window size W , providing the wBP performance with a reduced latency.

For both the wBP and the wTEP decoders, the performance between $W = 16$ and $W = 20$ does not significantly improve, and it is still far from the optimal BP and TEP values, included in the last row of each table. We have found that for $W \geq 50$ both decoders perform close to the not windowed versions, reducing the latency to a half. However, to keep the decoding latency low and further improve the performance, a more convenient solution is to increase M . In Table III, we include the wTEP performance for the $(3, 6, 100, 1024)$ ensemble. For the same latency, e.g. $W = 10$ for the $(3, 6, 100, 1024)$ code and $W = 20$ for the $(3, 6, 100, 512)$ code, a larger M leads to a quite improved WER performance. For equal WER, we conclude that the latency is very much reduced.

TABLE I
(3, 6, 100, 512) WER UNDER WBP DECODING.

W	$\epsilon = 0.45$	$\epsilon = 0.46$	$\epsilon = 0.47$
$W = 8$	0.7082	0.9943	1.0000
$W = 16$	0.0788	0.6301	0.9962
$W = 20$	0.047	0.6062	0.9944
$W = L + l - 1$ (BP)	0.0034	0.2732	0.9857

TABLE II
(3, 6, 100, 512) WER UNDER WTEP DECODING.

W	$\epsilon = 0.45$	$\epsilon = 0.46$	$\epsilon = 0.47$
$W = 8$	0.4769	0.9497	1.0000
$W = 16$	0.0234	0.3065	0.9470
$W = 20$	0.0209	0.3031	0.9276
$W = L + l - 1$ (TEP)	0.0002	0.0183	0.5876

V. CONCLUSIONS

For the BEC and finite-length LDPC codes, the TEP decoder exhibits a gain in performance compared to the BP, with linear complexity. In this letter we show that for LDPCC codes this

TABLE III
(3, 6, 100, 1024) WER UNDER WTEP DECODING.

W	$\epsilon = 0.46$	$\epsilon = 0.47$
$W = 10$	0.0490	0.6324
$W = 20$	0.0085	0.5543
$W = L + l - 1$ (TEP)	0.0003	0.0892

gain is achieved for much longer codes. This can be exploited either to improve the WER or the latency by reducing the code length. For a given target WER, under TEP decoding the code length is reduced by significantly decreasing the number of variables per LDPC code in the LDPCC. We also prove that TEP decoding can be successfully applied in a sliding window scheme, improving the performance of the windowed-BP or, alternatively, reducing the window size or the total code length. These results are of major interest, because practical LDPCC codes are limited by run-time complexity and decoding latency.

REFERENCES

- [1] A. J. Feltström and K. S. Zigangirov, "Time-varying periodic convolutional codes with low-density parity-check matrix," *IEEE Trans. on Information Theory*, vol. 45, no. 6, pp. 2181–2191, Sept. 1999.
- [2] T. Richardson and R. Urbanke, *Modern Coding Theory*. Cambridge University Press, Mar. 2008.
- [3] M. Lentmaier, A. Sridharan, D. J. Costello, and K. Zigangirov, "Iterative decoding threshold analysis for LDPC convolutional codes," *IEEE Trans. on Information Theory*, vol. 56, no. 10, pp. 5274–5289, Oct. 2010.
- [4] S. Kudekar, T. Richardson, and R. Urbanke, "Threshold Saturation via Spatial Coupling: Why Convolutional LDPC Ensembles Perform So Well over the BEC," *IEEE Trans. on Information Theory*, vol. 57, no. 2, pp. 803–834, Feb. 2011.
- [5] P. M. Olmos and R. Urbanke, "Scaling behavior of Convolutional LDPC ensembles over the BEC," in *2011 IEEE International Symposium on Information Theory Proceedings (ISIT)*, Aug. 2011, pp. 1816–1820.
- [6] D. Mitchell, A. Pusane, M. Lentmaier, and D. Costello, "Exact free distance and trapping set growth rates for LDPC convolutional codes," in *2011 IEEE International Symposium on Information Theory Proceedings (ISIT)*, Aug. 2011, pp. 1096–1100.
- [7] G. Corazza, A. Iyengar, M. Papaleo, P. Siegel, A. Vanelli-Coralli, and J. Wolf, "Latency constrained protograph-based LDPC convolutional codes," in *2010 6th International Symposium on Turbo Codes and Iterative Information Processing (ISTC)*, Sept. 2010, pp. 6–10.
- [8] P. M. Olmos, J. J. Murillo-Fuentes, and F. Pérez-Cruz, "Tree-structure expectation propagation for LDPC decoding over the BEC," *Submitted to IEEE Trans. on Information Theory*, 2011. [Online]. Available: <http://arxiv.org/abs/1201.0715>
- [9] P. Olmos, J. J. Murillo-Fuentes, and F. Pérez-Cruz, "Tree-structured expectation propagation for decoding finite-length LDPC codes," *IEEE Communications Letters*, vol. 15, no. 2, pp. 235–237, Feb. 2011.
- [10] R. Koetter and F. Kschischang, "Coding for errors and erasures in random network coding," *IEEE Trans. on Information Theory*, vol. 54, no. 8, pp. 3579–3591, Aug. 2008.
- [11] A. Thangaraj, S. Dihidar, A. Calderbank, S. McLaughlin, and J. Merolla, "Applications of LDPC codes to the wiretap channel," *IEEE Trans. on Information Theory*, vol. 53, no. 8, pp. 2933–2945, Aug. 2007.
- [12] A. Dana, R. Gowaikar, R. Palanki, B. Hassibi, and M. Effros, "Capacity of wireless erasure networks," *IEEE Trans. on Information Theory*, vol. 52, no. 3, pp. 789–804, March 2006.
- [13] A. Iyengar, M. Papaleo, P. H. Siegel, J. K. Wolf, A. Vanelli-Coralli, and G. E. Corazza, "Windowed decoding of protograph-based LDPC convolutional codes over erasure channels," *IEEE Trans. on Information Theory*, no. 99, p. 1, 2011.
- [14] J. Thorpe, "Low density parity check (LDPC) codes constructed from protographs," JPL IPN Progress Report 42-154, Tech. Rep., 2003.