Tree-Structured Expectation Propagation for Decoding Finite-Length LDPC Codes

Pablo M. Olmos, Juan José Murillo-Fuentes, and Fernando Pérez-Cruz

***Abstract*—In this paper, we propose Tree-structured Expectation Propagation (TEP) algorithm to decode finite-length Low-Density Parity-Check (LDPC) codes. The TEP decoder is able to continue decoding once the standard Belief Propagation (BP) decoder fails, presenting the same computational complexity as the BP decoder. The BP algorithm is dominated by the presence of stopping sets (SSs) in the code graph. We show that the TEP decoder, without previous knowledge of the graph, naturally avoids some fairly common SSs. This results in a significant improvement in the system performance.**

***Index Terms*—Tree-structured expectation propagation, LDPC decoding, finite-length analysis.**

I. Introduction

TREE-structured expectation propagation (TEP) [1] has been proposed by *Olmos et al.* in [2] as a decoding algorithm for low-density parity-check (LDPC) codes over discrete memoryless channels (DMCs). The TEP improves the performance of the belief propagation (BP) algorithm thanks to a better estimation of the posterior probability for each bit in the codeword. In [2], the authors focus on the TEP limiting performance over the binary erasure channel (BEC). It is shown that the TEP asymptotically decodes up to a higher fraction of errors than the BP for regular and irregular LDPC codes. In addition, for the BEC, the TEP complexity equals the BP, unlike other proposals to improve the BP estimate such as the generalized-BP (GBP) [3] .

For infinite-length codes, the analysis of the decoding performance of the BP and the TEP is based on the absence of cycles in the graph along the decoder iterations [4], [2]. This assumption does not hold for finite-length codes, where some patterns of errors yield a set of cycles that prevent the variables involved to be decoded. These structures of cycles, where the BP cannot find an unique solution, are known as stopping sets (SSs). For practical short-length LDPC codes the SSs limit the performance of the decoders. In this paper, we analyze the decoding capabilities of the TEP compared to the BP for finite-length codes over the BEC. We show that some SSs of the BP are broken by the TEP decoder, thus improving appreciably the decoding process. The experimental results for regular and irregular LDPC codes show that the TEP decoder significantly outperforms the BP decoder in the waterfall region [5], where the BP performance is dominated by large SSs.

Manuscript received October 22, 2010. The associate editor coordinating the review of this letter and approving it for publication was S. Yousefi.

This work was partially funded by the Spanish government (Ministerio de Educación y Ciencia 2009-14504-C02-01,02, Consolider-Ingenio 2010 CSD2008-00010), Universidad Carlos III (CCG10-UC3M/TIC-5304) and the European Union (FEDER).

P. M. Olmos and J. J. Murillo-Fuentes are with the Dept. Teoría de la Señal y Comunicaciones, Escuela Técnica Superior de Ingeniería, Universidad de Sevilla, Paseo de los Descubrimientos s/n, 41092 Sevilla, Spain (e-mail: {olmos, murillo}@us.es).

F. Pérez-Cruz is with the Dept. Teoría de la Señal y Comunicaciones, Universidad Carlos III de Madrid, Spain (e-mail: fernando@tsc.uc3m.es).

Digital Object Identifier 10.1109/LCOMM.2011.03.102015

II. BP and TEP for the BEC

The BP algorithm [6] is the basic tool for LDPC decoding. Although it was originally proposed as a message passing algorithm, for the BEC it exhibits an alternative and simpler formulation, in which the non-erased variable nodes are removed from the graph after each iteration. The BP, under this interpretation, is referred to as the *peeling decoder* [7]. We briefly introduce the peeling decoder since the TEP decoder for the BEC is an extension of it. Both algorithms are easily described using the Tanner graph of the LDPC code [4]. The graph has n variable nodes $V_1, \ldots, V_n$ and $n(1-r)$ check nodes $P_1, \ldots, P_{n(1-r)}$, where r is the rate of the code. The degree of a variable or check node is defined as the number of edges connected to it.

A. Peeling decoder

The decoder is initialized by removing from the graph all the variable nodes corresponding to non-erased bits. We also remove all the connections from these variable nodes. After removing a variable node whose value was one, we change the parity of the check node(s) it was connected to. The BP algorithm proceeds by removing a check node and a variable node in each iteration:

1) It looks for any check node linked to a single variable node, i.e., a check node of degree one. The BP decoder copies the parity of this check node into the variable node and removes the check node.
2) Second, it removes the variable node that we have just de-erased. If the variable was a one, it changes the parity of the check node(s) it was connected to.
3) It repeats Steps 1) and 2) until all the variable nodes have been removed, successfully finishing the decoding of the received word, or until there are no degree-one check nodes left, yielding an unsuccessful decoding.

B. TEP decoder for BEC

The TEP decoder works over the LDPC graph using not only the check nodes of degree one but also the degree-two check nodes. A check node of degree two tells us that the variable nodes connected to it are either equal, if the check has parity zero, or opposite, otherwise. The decoder is initialized like the peeling decoder, removing all the bits

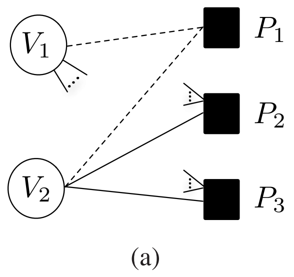

Fig. 1. In (a) we show two variable nodes, V_1 and V_2, that share a check node of degree two. In (b), V_1 heirs the connections of V_2 (solid lines) and P_1 and V_2 are removed.

that have not been erased by the channel. In each iteration, the TEP performs as follows:

1) It looks for a check node of degree one or two.
2) If a check of degree one is found, the TEP recovers the associated variable, performing the steps 1) and 2) of the peeling decoder in Section II-A.
3) If a check of degree two is found, the decoder removes the check node of degree two from the graph together with one of the variable nodes connected to it and the two associated edges. Then it reconnects to the remaining variable node all the check nodes that were connected to the removed variable node. The parities of the check nodes re-connected to the remaining variable node have to be reversed if the removed check node had parity one.
4) Steps 1)-3) are repeated until all the variable nodes have been removed or the graph runs out of check nodes of degree one or two.

The step 3) 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. V_2 is recovered once V_1 is de-erased. Finally, if P_1 is parity one, the parities of P_2 and P_3 are reversed.

The TEP removes a check and a variable node per iteration, as the BP does. Hence, we are able to improve the decoding performance without increasing the complexity. When a check node of degree two is removed, the value of the removed variable node remains unknown. This variable becomes known, when the remaining variable node in the graph has been decoded. In [2], it is also proved that the TEP solution does not depend on the order in which check nodes of degree one or two are removed.

C. Stopping sets for the TEP decoder

The decoding performance for finite length codes, as discussed in the introduction, is dominated by the presence of SSs [4], [5]. In Fig. 2(a) and (b), we include an example that shows why the TEP decoder is able to improve the BP performance for finite-length codes. In Fig. 2(a), the subgraph in thick solid lines with variable nodes V_2, V_4 and V_6 constitutes a SS for the BP decoder. If the three variables are erased, the BP decoder cannot recover them. Nevertheless, the TEP decoder can decode this subgraph: the check node P_2 is degree two and is eliminated along with V_6 following the step 3) of the TEP in Section II-B. Variable V_4 heirs the connections of V_6 and, hence, it gets doubly connected to P_6. In practice, these

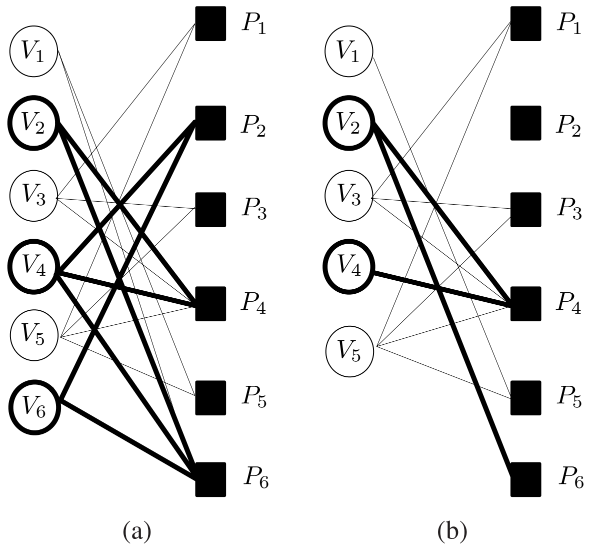

Fig. 2. In (a), the variables V_2, V_4 and V_6 form one SS (thick solid line). In (b), we show the graph once the TEP has removed P_2 and V_6.

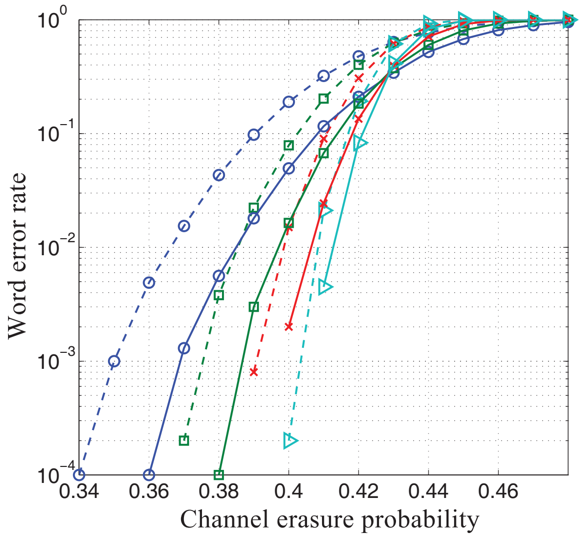

Fig. 3. TEP (solid line) and BP (dashed line) decoding performance for a regular LDPC (3,6) code with code lengths $n = 2^8$ ($\circ$), $n = 2^9$ ($\square$), $n = 2^{10}$ ($\times$) and 2^{11} ($\triangleright$).

two edges can be removed. In Fig. 2(b), we plot the graph after this process. We can see that P_6 becomes degree one, which makes V_2 recoverable.

We have demonstrated the existence of a SS that the TEP successfully decodes. Any SS is decodable by the TEP if Step 3) of the TEP decoder breaks its cycles and reveals some of their variables.

III. Experimental Results

We first consider a rate $1/2$ regular $(3,6)$ LDPC code. In Fig. 3, we plot the word error rate (WER) obtained with the TEP (solid lines) and the BP (dashed lines) decoders with code lengths $n = 2^8$ ($\circ$), $n = 2^9$ ($\square$), $n = 2^{10}$ ($\times$) and 2^{11} ($\triangleright$). We have averaged the results between 100 codes samples and 10^5 codewords for each sample. Note that the TEP improves the BP decoder in all cases but the gain is more significant for short codes.

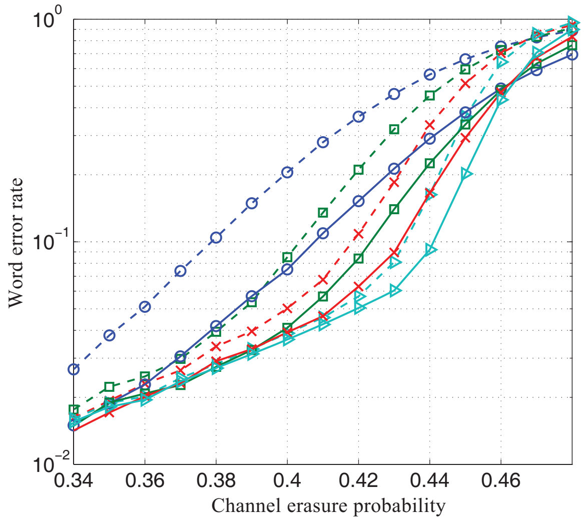

Fig. 4. TEP (solid line) and BP (dashed line) decoding performance for the irregular LDPC code defined by (1) with code lengths $n = 2^8$ ($\circ$), $n = 2^9$ ($\Box$), $n = 2^{10}$ ($\times$) and 2^{11} ($\triangleright$).

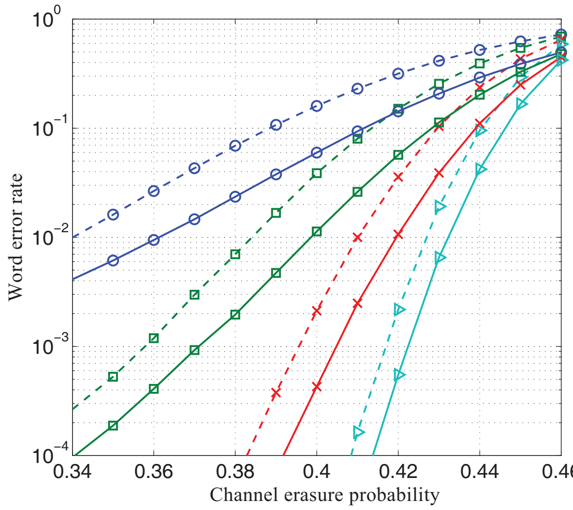

Fig. 5. TEP (solid line) and BP (dashed line) decoding performance for the irregular LDPC code defined by (1) with code length/expurgation parameters of $n/s_\epsilon = 2^8/10$ ($\circ$), $n/s_\epsilon = 2^9/20$ ($\Box$), $n/s_\epsilon = 2^{10}/40$ ($\times$) and $n/s_\epsilon = 2^{11}/80$ ($\triangleright$).

Capacity achieving LDPC codes for the BP are defined by irregular distributions. These codes have higher limiting thresholds, but their performance for finite-lengh codes can be poor, specially in the error floor region [5]. On the contrary, for the regular $(3, 6)$ code in the former example, we can easily find samples with no error floor. We have worked with a rate $1/2$ irregular code defined by the next node degree distributions [4]:

$$\begin{aligned}L(x) &= 0.6253x^2 + 0.1663x^3 + 0.0752x^4 + 0.0421x^5 \\ &\quad + 0.0266x^6 + 0.018x^7 + 0.0132x^8 + 0.01x^9. \\ &\quad + 0.0078x^{10} + 0.0063x^{11} + 0.0050x^{12} + 0.0042x^{13}, \\ R(x) &= x^6. \end{aligned} \tag{1}$$

In Fig. 4 we plot the results for code lengths of $n = 2^8$ ($\circ$), $n = 2^9$ ($\Box$), $n = 2^{10}$ ($\times$) and 2^{11} ($\triangleright$). Each curve has been averaged with 20 code samples and 10^5 codewords. In the waterfall region, where the error probability falls off sharply, the TEP decoder (solid curves) improves the BP error probability (dashed lines) in all cases, but specially for short codes. It is known that the performance in the waterfall region is determined by large SSs [4]. At the light of the results in Fig. 3 and Fig. 4, the TEP decoder is clearly more efficient than the BP to avoid these kinds of SSs.

In Fig.4, we can appreciate that both the TEP and BP present similar error floor, characterized by an slow decrease of the error probability with ϵ. Hence, the low weight SSs similarly degrade the TEP and BP decoders performance. In practice, the error floor region can be greatly reduced by modifying the LDPC code, see [8], [5] for example. Using these techniques, we are able to expurgate all the short cycles in the LDPC code and, therefore, avoid short SSs and decoding failures with weight below an *expurgation parameter* s_ϵ [4]. In Fig. 5, we illustrate the effect of the expurgation in the irregular code in (1). We plot the TEP and BP curves for the expurgated code with code length/expurgation parameters of $n/s_\epsilon = 2^8/10$ ($\circ$), $n/s_\epsilon = 2^9/20$ ($\Box$), $n/s_\epsilon = 2^{10}/40$ ($\times$) and $n/s_\epsilon = 2^{11}/80$ ($\triangleright$). We can better observe now the TEP gain in the whole range of the erasure probability. The TEP decoder then provides a significant coding gain, not only for regular LDPC codes, but also for capacity achieving irregular LDPC codes.

IV. Conclusions

In this paper we analyze the performance of the TEP decoder for practical finite-length regular and irregular LDPC codes. We describe the TEP for the BEC to show that the new additional step introduced to the peeling decoder allows the successful decoding of a SS with the same computational complexity. In the numerical experiments included, the TEP exhibits a remarkable improvement in the waterfall region. These results are of major interest, as a first step in the design of good finite-length codes for the TEP to exploit the gain introduced.

References

[1] T. Minka and Y. Qi, “Tree-structured approximations by expectation propagation,” in *Proc. Neural Information Processing Systems Conference (NIPS)*, 2003.

[2] P. M. Olmos, J. J. Murillo-Fuentes, and F. Pérez-Cruz, “Markov chain expectation propagation for decoding in erasure channels,” *IEEE Trans. Inf. Theory*, submitted.

[3] J. Yedidia, W. Freeman, and Y. Weiss, “Constructing free-energy approximations and generalized belief propagation algorithms,” *IEEE Trans. Inf. Theory*, vol. 51, no. 7, pp. 2282–2312, July 2005.

[4] T. Richardson and R. Urbanke, *Modern Coding Theory*. Cambridge University Press, 2008.

[5] C. Di, D. Proietti, T. Richardson, E. Telatar, and R. Urbanke, “Finite length analysis of low-density parity-check codes on the binary erasure channel,” *IEEE Trans. Inf. Theory*, vol. 48, no. 6, pp. 1570–1579, June 2002.

[6] D. J. C. MacKay and R. M. Neal, “Near Shannon limit performance of low density parity check codes,” *IEE Electron. Lett.*, vol. 32, no. 18, pp. 1645–1646, 1996.

[7] T. Richardson and R. Urbanke, “The capacity of low-density parity check codes under message-passing decoding,” *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 599–618, Feb. 2001.

[8] X. Zheng, F. C. M. Lau, and C. K. Tse, “Constructing short-length irregular LDPC codes with low error floor,” *IEEE Trans. Commun.*, vol. 58, no. 10, pp. 2823–2834, Oct. 2010.