Modified augmented belief propagation for general memoryless channels

Benjamin Gadat, Charly Poulliat

Modified augmented belief propagation for general memoryless channels

Benjamin Gadat
Airbus Defense and Space,
Signal and Algorithm Dpt, Toulouse, France
Email: benjamin.gadat@airbus.com

Charly Poulliat
University of Toulouse, IRIT/INP-ENSEEIHT/CNRS,
Toulouse, France
Email: charly.poulliat@enseeiht.fr

Abstract—**In this paper, we propose an efficient implementation of the augmented belief propagation (ABP) algorithm for low-density parity-check codes over general memoryless channels. ABP is a multistage BP based decoder that uses a backtracking processing when decoding fails. The algorithm proceeds in two main steps, namely a symbol selection step and an *augmented* decoding step. The former is based on a criterion related both to the stopping subgraph connectivity and to the input reliability, while the latter can be either implemented using a list based or a greedy approach. Compared to the original implementation, we consider a different approach for both steps. First, the proposed node selection is only based on the dynamic of sign changes of the extrinsic messages at the variable nodes output. This enables us to consider indifferently general memoryless channels, while still taking into account the graph irregularity. Then, we propose a simple yet efficient implementation of the augmented decoding procedure based on pruning of the branching tree The proposed algorithm shows near maximum likelihood decoding performance while decreasing the overall complexity (computation and memory) of the original algorithm. Moreover, complexity-performance trade-off is an built-in feature for this kind of algorithm.**

I. Introduction

Belief propagation (BP) decoder of sparse graph codes performs iterative message passing on a code graph. This is a sub-optimal approach that assumes extrinsic messages independence which is a good approximation only if the graph is large enough, leading to near optimal detection. However, for short codeword block lengths, the presence of cycles in low-density parity-check (LDPC) code graphs can introduce a noticeable loss in performance when compared to the maximum likelihood (ML) decoding performance. One possible manner to efficiently address this problem is to combine efficiently iterative BP decoding with the reliability based ordered statistic decoding algorithm [1]. However, despite of its very good performance, it suffers from a heavy computational complexity that makes this approach difficult to use in practice. [2], [3] proposes a new method to improve the performance in the error floor region. This approach, referred to as augmented belief propagation (ABP), is a generalization of the approach that has been proposed initially by [4]. ABP is a multistage BP based decoder that uses a well-thought backtracking processing when decoding fails. The algorithm iteratively proceeds in two main steps, namely a symbol selection step and an *augmented* decoding step. The former is mainly based on a criterion related to both the stopping subgraph connectivity and to the input reliability of the received information, while the latter can be either implemented using a list based or a greedy decoding approach. The main idea is very close in spirit to the Maxwell decoder used in [5]. Indeed, in these approaches, when the decoding algorithm fails, the value of a bit is guessed and then the decoding procedure continues until a new decoding failure. By analyzing the different possible decoding trajectories, one expect to finally converge to the right codeword when applying a well-thought variable node selection strategy. There are several other approaches (see for example [6] and references therein) that can be related to ABP.

In this paper, we propose an efficient implementation of the ABP algorithm for LDPC codes over general memoryless channels. First, we consider a node selection that is only based on the dynamic of sign changes of the extrinsic messages at variable nodes output. This enables us to consider indifferently general memoryless channels, while still taking into account the influence of irregularity in the graph connectivity. Then, we propose a simple yet efficient implementation of the augmented procedure based on an efficient pruning of the branching tree used for the decoding schedule with decoder reinitialization. The proposed algorithm show near maximum likelihood decoding performance while decreasing the overall complexity (computation and memory) of the original algorithm. Moreover complexity-performance trade-off is an built-in feature for this kind of algorithm. The paper is organized as follows. Section II gives some notations and definitions. Section III introduces the ABP algorithm and section IV presents our proposed implementation. Then, simulation results are given in Section V and conclusions are drawn in Section VI.

II. Notations and Definitions

Binary LDPC codes are linear block codes defined over $\mathbb{F}_2$. A LDPC code C_H is usually defined by a sparse parity-check matrix $\mathbf{H}$ of size $M \times N$, where N is the codeword length, $M \geq N - K$ the number of parity check equations and K is information length in bits. Using this LDPC code, an information message vector $\mathbf{u} = [u_0, \cdots, u_{K-1}] \in \mathbb{F}_2^K$ is encoded into a codeword $\mathbf{z} = [z_0, \cdots, z_{N-1}] \in F_2^N$ that belongs to the null space of $\mathbf{H}$, ie. $\mathbf{H}\mathbf{z}^\top = \mathbf{0}$ where $^\top$ holds for transposition [7]. The parity-check matrix $\mathbf{H}$ can be represented by its corresponding Tanner graph $\mathcal{G}$ [8]. A Tanner graph is a bipartite graph consisting in two sets of nodes: the variable nodes associated with the codeword bits (columns of $\mathbf{H}$) and the check nodes associated with the parity check constraints (rows of $\mathbf{H}$). An edge joins a variable nodes (VN) v_n to a check nodes (CN) c_m if $\mathbf{H}(m,n) = 1$. For a given parity $\mathbf{H}$, denote the set of variable nodes by $\mathcal{V}$, the set of check nodes by $\mathcal{C}$ and the set of edges by $\mathcal{E}$. Then, $\mathcal{G}$ is completely described by the triple $(\mathcal{V},\mathcal{C},\mathcal{E})$.

The codewords $\mathbf{z}$ are then sent over a binary memoryless channel. Let $\mathbf{y} = [y_0, \cdots, y_{N-1}]$ be the received vector at the channel output. At the receiver, belief propagation decoding (or one of its relatives) is used to perform iterative message passing on the code graph assuming local independence of the messages transiting on the edges of the graph. Without loss of generality, we will consider log-likelihood ratio (LLR) based BP decoding for the ease of exposition of the proposed algorithm. The LLR-based BP decoding iteratively exchanges LLR messages between variables and check nodes which are processed locally at each node. From the observations $\mathbf{y}$, we can compute the expression of the initial LLR values from the channel observations, noted as:

$$L_n^{ch} = \log\left(\frac{p(z_n = 0|y_n)}{p(z_n = 1|y_n)}\right).$$

Then, the following update rules are applied iteratively at each iteration ℓ [7]:

- **Iterations:** computation of extrinsic messages from variable node v_n (resp. check node c_m) to check node c_m (resp. variable node v_n)

$$L^\ell_{n\to m} = L_n^{ch} + \sum_{m'\in\mathcal{M}(n)\backslash m} \tilde{L}^{\ell-1}_{m'\to n}$$

$$\tilde{L}^\ell_{m\to n} = 2\tanh^{-1}\left(\prod_{n'\in\mathcal{N}(m)\backslash n} \tanh\left(L^\ell_{n'\to m}/2\right)\right)$$

  where $\mathcal{M}(n)\setminus m$ (resp. $\mathcal{N}(m)\setminus n$) is the set of check nodes (resp. variable nodes) connected to v_n(resp. c_m) excluding the m-th check node (resp. the n-th variable node) and $\tilde{L}^\ell_{m'\to n}$ (resp. $L^\ell_{n'\to m}$) is the extrinsic message from the check node $c_{m'}$ (resp. the variable node $v_{n'}$) to the variable node v_n (resp. the check node c_m).

- **A posteriori LLR computation:** $\forall n \in [0, N-1], \forall\ell$

$$L_n^{app,\ell} = L_n^{ch} + \sum_{m'\in\mathcal{M}(n)} \tilde{L}^\ell_{m'\to n}.$$

  The estimated codeword bit is then given as

$$\hat{z}_n = \frac{1 - \mathrm{sign}(L_n^{app,\ell})}{2}.$$

At the first iteration, we have $L^1_{n\to m} = L_n^{ch}, \forall m, n$. The above iterative procedure is run until a codeword has been detected (ie $\mathbf{H}\hat{z}^\top = \mathbf{0}$) or the maximum number of iterations, noted L_0, has been reached. In this latter case, a decoding failure is declared (ie. an error has been detected). In case of decoding failure, denote $\mathcal{C}_{SUC}$ the set of unsatisfied check (SUC) nodes when evaluating the syndrome $\mathbf{s} = \mathbf{H}\hat{z}^\top$, $\mathcal{V}_{SUC}$ the set of variable nodes that are connected to $\mathcal{C}_{SUC}$ by at least one edge in G, and $\mathcal{E}_{SUC}$ the set of edges connecting the nodes in $\mathcal{V}_{SUC}$ with nodes in $\mathcal{C}_{SUC}$ [2]. The set $\mathcal{C}_{SUC}$ induces an extracted subgraph $G_{SUC} = (\mathcal{V}_{SUC},\mathcal{C}_{SUC},\mathcal{E}_{SUC})$. Note that the variable node degree of $v_n \in \mathcal{V}_{SUC}$ with respect to G_{SUC} is less or equal to the variable node degree of $v_n \in V$ with respect to the original graph G. This type of structure is usually the core of ABP decoders or BP decoder with backtracking [2].

As we will see in the following sections, another variable of interest to track is the number of sign changes of extrinsic messages (ie. extrinsic message oscillations) from variable nodes to check nodes along iterations [9]. Let $S^\ell_{n\to m}$ be the number of sign changes of the extrinsic message from variable node v_n to check node c_m up to iteration ℓ and S^ℓ_n the total sign changes of all extrinsic messages at the output of variable node v_n up to iteration ℓ. Then, we have $\forall n, m$

$$S^\ell_{n\to m} = \begin{cases} S^{\ell-1}_{n\to m} & \text{if } \mathrm{sign}(L^\ell_{n\to m}) = \mathrm{sign}(L^{\ell-1}_{n\to m}) \\ S^{\ell-1}_{n\to m} + 1 & \text{otherwise} \end{cases}$$

$$S^\ell_n = \sum_{m\in\mathcal{M}(n)} S^\ell_{n\to m}. \qquad (1)$$

By convention, we have $S^1_{n\to m} = 0$.

III. Augmented Belief Propagation

A. Main features

Augmented belief propagation is a multistage BP based decoder that uses a well-thought backtracking processing when decoding fails [2], [10]. The ABP algorithm relies first on applying the classical BP algorithm for the first L_0 iterations. If decoding is successful, then the iterative decoding procedure stops. However, if the BP decoding fails (ie. no codeword has been decoded), the ABP procedure is used. The ABP decoding procedure performs a multistage decoding using a two steps process that can be summarized as follows: at each stage j a variable node v_n that is likely to be in error is selected and its initial message L_n^{ch} is replaced by a *saturated* message, ie. $L_n^{ch} = \pm S_{\max}$. Then, a tentative BP decoding is applied for L_1 additional iterations on both possible values $+S_{\max}$ or $-S_{\max}$, defining the augmented decoding procedure. Thus, the two main features of this kind of algorithm are: (a) the node selection process and (b) the augmented decoding procedure that consists in an efficient binary testing based decoding schedule.

B. ABP decoding procedure

The overall procedure can described as follows. In the following, we present both a parallel and a sequential decoding schedule as initially proposed in [2]. It can be represented by a branching tree as given in Fig. 1 and Fig. 2 for a decoding depth $j_{\max} = 3$.

The following procedure can be used with both scheduling. First, L_0 BP decoding iterations are performed. If decoding fails, a variable node v_n^0 is selected according to a *selection* algorithm. Then, the first stage ($j = 1$) of the augmented procedure is applied: two decoding attempts can be performed at this stage by setting $L_n^{ch} = -S_{\max}$ or $L_n^{ch} = +S_{\max}$. Then, L_1 iterations of BP decoding are performed for each possible bit correction value. If the decoder finds a codeword, it stores it and continues. In its original version, it is assumed that the initial messages are stored before applying the two step decoding testing (incoming messages from check nodes

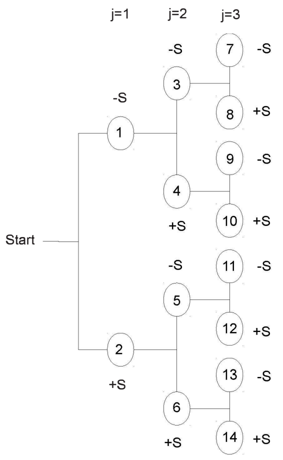

Fig. 1. Parallell Scheduling

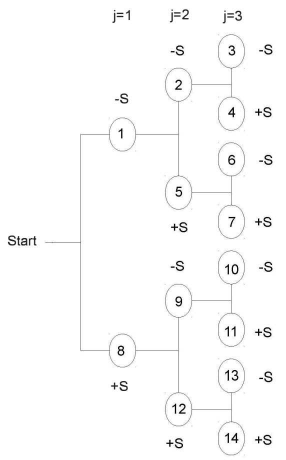

Fig. 2. Sequential Scheduling

to variable nodes). We can see that this augmented decoding procedure aims to explore two possible decoding trajectories with a guessed value of the variable node v_n^0, which is very close in spirit to the Maxwell decoder used in [5]. At the next stage $j = 2$, the same two-steps process is applied for the two preceding decoding trajectories following the testing order as given in Fig. 1. For example, for the decoding trajectory at stage $j = 1$ considering v_n^0 with $L_n^{ch} = -S_{\max}$, a new symbol v_n^1 is selected and messages along the edges of the graph are stored for this configuration. Then the same decoding testing procedure is applied considering the two possible saturated values for v_n^1 and running additional L_2 iterations. The same procedure is applied for the decoding trajectory with initial message v_n^0 with $L_n^{ch} = +S_{\max}$. Thus at each stage, 2^j additional decoding trajectories can be tested. The procedure continues until the maximum number of allowed stages $j_{\max}$ is reached.

In its original (and full complexity) version, ABP needs to store at each step the selected nodes (to be saturated) and all the messages along the edges in the graph (from check nodes to variable nodes) that are required for a configuration to be tested. However, lower memory complexity can be achieved by restarting decoding after each node selection, avoiding storage of messages in the graph. During the parallel testing procedure, it is also assumed that all valid codewords are collected and then list decoding is performed.

C. Implementation issues

Apart from the memory complexity, it clearly appears that there are two main aspects for this kind of algorithm that can be optimized: (a) the node selection strategy and (b) the testing schedule and decoding strategy, ie. how to explore efficiently the branching tree and what is the best decoding strategy. For both aspects, several propositions have still been explored. In [2], an improved version of the node selection initially given in [4] is proposed. When decoding fails, the nodes are selected from the graph $\mathcal{G}_{SUC}$ induced by non satisfied check nodes. To select a variable node from $\mathcal{G}_{SUC}$, they have shown heuristically that the best candidate is among variables nodes with highest degrees in $\mathcal{G}_{SUC}$ and with the lowest channel information reliability. To decrease the decoding complexity, they have also proposed a greedy approach that terminates exploration of the branching tree as soon as a valid codeword is reached. To reduce the memory requirement, they have also proposed a sequential testing schedule for the branching tree (as given in Fig. 2) that corresponds to a depth-first search while the parallel testing is a breadth first search approach. The complexity-performance trade-off is addressed through different parameters such as the number of allowed stages, the number of initial iterations L_0 and the number of allowed additional iterations per stage.

By applying ABP on state of the art codes, [2] showed that ABP is a very efficient method that can approach maximum likelihood performance if sufficient complexity is allowed. Moreover it can perform the same than other algorithm such as BP combined with ordered statistic decoding but with a reduced complexity [2]. For all these aspects, ABP is for sure a very attractive decoding method.

However, the original method has some weaknesses :

- First, the method can be only applied to channels enabling information reliability that is computed from channel observations. The proposed method cannot be applied to channels like the binary symmetric channel (BSC) due to the proposed node selection criterion;
- If memory requirement can be drastically softened by enabling to restart the decoding (with, in some cases, only a slight decrease in performance [2]), the node selection process implies to extract dynamically at each selection step the subgraph G_{SUC} that can be different from a stage to another. So the need for extracting information from local substructures when decoding fails increases the complexity of the selection step;
- For the testing schedule and decoding step, it would be interesting to see if some efficient pruning of the branching tree could be performed and if some stopping criterion could be found that would enable to have an algorithm with performance close to the full list decoding approach while being less complex.

IV. Proposed Implementation

In this section, we give a description of the proposed implementation of an ABP procedure, which copes with the previously stated weaknesses:

- it can be applied to general memoryless channels including channel without considering extraction of graph substructures for node selection;
- it implements a pruning method of the branching tree while introducing an objective stopping criterion;
- it naturally takes into account irregularity in the Tanner graph of the code.

A. Node selection

The main idea of the proposed implementation is as follows: we do not rely on channel observations for the node selection process. A different approach is preferred that relies on the dynamic of the *extrinsic* LLR messages at variable

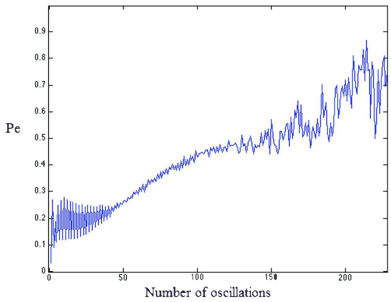

Fig. 3. Probability of variable node error wrt. number of extrinsic oscillations.

nodes output. As observed by [9], [11], oscillations in the sign of LLR messages (both a priori and extrinsic) greatly influence the behavior of the decoding trajectories both in the waterfall and in the error floor regions. In [11] [9], it is shown that observed oscillating messages are closely related to erroneous bits. An "Oscillations" based ordered statistic decoder (OSD) - BP algorithm is proposed where the codeword bit sorting step of the OSD algorithm is performed based on the number of sign oscillations of the *a posteriori* LLR messages. Direct application of this criterion to the ABP node selection step has been proposed in [10].

In this paper, we propose to use the total oscillations of the **extrinsic** messages for a given variable node as a reliability information (cf. equation (1)). The rationale behind is as follows: an highly oscillating variable node will prevent the decoder to converge due to the high induced dynamic during the decoding process [12], [9]. By fixing the value of those nodes, we expect to mitigate their influence on the decoding process. Counting the sign changes on a edge perspective allow us to take into account the degree of connectivity of a variable node. This property is illustrated in Fig. 3 where the error probability of a node is plotted versus the total number of extrinsic message oscillations when BP decoding is applied to the Tanner code $(155, 64)$ at $E_b/N_0 = 2$ dB. As we can see, the probability to be in error is relatively high if the extrinsic messages have often oscillated. As illustrated latter, a node selection based only on the oscillation of the a posteriori messages is only efficient for regular codes as it is considered in [10]. The proposed criterion takes into account two properties such as the degree of connectivity of a variable node and an information of reliability based on total oscillations of extrinsic messages. So at each node k of the decoding tree, if the decoding fails, we select the variable node having the highest number of oscillations for the extrinsic messages, according to Eqn. (1), among variable nodes that have not been selected up to that step.

B. *Testing schedule and decoding algorithm*

For the decoding/testing schedule, we propose to use the sequential schedule as proposed in the original algorithm, ie. performing a depth-first search in the branching tree to benefit from a memory complexity decrease of a depth-first search compared to a breadth-first search. However, we can still improve the efficiency of the depth-first search by considering the two following observations:

- **Observation 1:** if, at a given stage j, a valid codeword is found, we can store it and prune all the paths at higher stages that can be explored from that branch before continuing our search. For example, considering that the depth-first search has converged to a codeword after stage $j = 1$ associated with a saturated observation value $-S_{\max}$ of the variable node v_n^0. All nodes in the branching tree from node 2 to node 7 can be ignored and the search continues with node 8 (see Fig. 2).
- **Observation 2:** If we know the minimum distance $d_{\min}$ of the LDPC code (or a rough estimate), we can apply a bounded distance stopping criterion. Indeed, if at a given stage, the decoder converges to a valid codeword while having corrected less than $\lfloor d_{\min} - 1 \rfloor / 2$ codeword positions, then we can stop the search and output the corresponding codeword.

Based on these observations, we apply the sequential decoding using a list based decoding algorithm and applying both the pruning heuristic and the proposed stopping criterion. Moreover, to enable an efficient exploration of the decoding tree, a branch selection is performed. At each node k, if decoding fails and a node is selected, the subsequent branch of the tree to be explored is based on the opposite sign of the APP decision message of the selected node. At the end of the augmented decoding procedure, if the list of codewords contains more than one candidate, a maximum likelihood decoding rule is applied. Thus, an efficient modified version of the list decoding ABP algorithm A from [2] can be implemented.

V. Simulation results

To illustrate the influence of the choice of a node selection based on a posteriori oscillations or based on extrinsic messages, we investigate the performance of our modified ABP algorithm with irregular codes. To do so, we consider a moderate and high rate SeIRA codes [7] on the fast fading Rayleigh channel with 50 initial BP decoding iterations and 10 additional iterations per additional stage. As we can see in Fig 4 there is a significant improvement of the performance in the error floor region. It has not been noted in [10] since the code that have been used are regular codes. The rational behind is that taking into account the total oscillations of extrinsic messages enables us to take into account the irregularity of a variable node in the selection node step. We now consider the $(155, 64)$ Tanner code [2] on the AWGN channel. In Fig. 5, we compare our oscillations based ABP (OABP) with the original ABP algorithm A (full complexity version) and B (greedy version) from [2] with $L_0 = 100$ and additional iterations at each step $L = 10$. As shown in the figure, OABP performs almost the same than the ABP algorithm B while being very close to the ABP algorithm A performance for 11 layers and operating close to the ML performances. For less layers, OABP seems more efficient than the ABP algorithm B from [2]. The main advantage of our approach in this case is the low complexity induced by systematic reset of the decoder messages and the use of a more simple selection node criterion (no need for subgraph extraction). Another strength of the proposed approach is its ability to cope with non "soft" channel such as the BSC, which is not the case when using the node selection as in [2]. In Fig. 6, we compare the performance

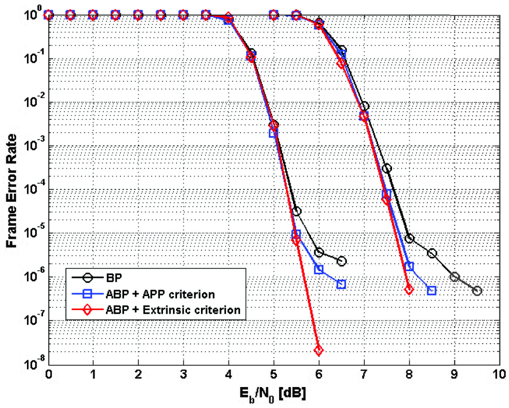

Fig. 4. Comparison of node selection for $R = \{2/3, 4/5\}$ SeIRA codes

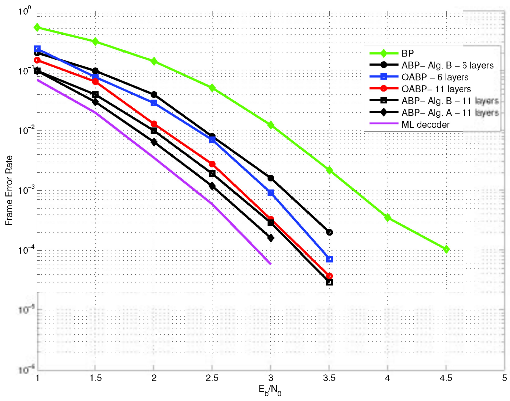

Fig. 5. Comparison of ABP algorithm A and B from [2] for Tanner code.

of our OABP on the BSC with the classical BP decoder. It can be also compared to the results of the FAID decoder with decoder diversity (9239 FAID decoders) in [13]. As a proof of concept, to show that ML performance can be achieved, we set $L_0 = 1000$ and additional number of iterations is also set to $L = 1000$ with 11 layers. The curves shown in Fig. 6 are very close to the ML bound given in [13] but with a lower complexity than the FAID decoder diversity using 9239 decoders. The same behavior is observed for the $N = 2640$ $R = 1/2$ Margulis code as shown in Fig. 6. The number of iterations is set to 100 with 10 additional iterations.

VI. Conclusion

In this paper, we have proposed an efficient implementation of the augmented belief propagation algorithm for LDPC codes over general memoryless channels. Compared to the original implementation, we have considered a different approach for both steps. First, we have shown that a good choice for the node selection is to track the dynamic of sign changes of the extrinsic messages at the output of the variable nodes. This enables us to consider indifferently general memoryless channels, such as the BSC or the AWGN channel, while still taking into account the influence of irregularity in the graph connectivity. Then, we propose a simple yet efficient implementation of the augmented decoding procedure based on a pruning of the branching tree used for the sequential decoding schedule with decoder reinitialization. The proposed algorithm shows near ML decoding performance while decreasing the overall complexity of the original algorithm. Moreover complexity-performance trade-off is an built-in feature for this kind of algorithm. Future works will consider the application to more general message passing algorithms.

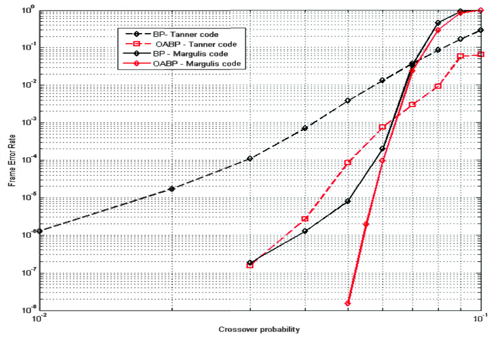

Fig. 6. OABP versus BP on the BSC channel for the Tanner code.

References

[1] M. Fossorier, "Iterative reliability-based decoding of low-density parity check codes," *IEEE J. on Sel. Areas in Comm.,*, vol. 19, no. 5, pp. 908–917, May 2001.

[2] N. Varnica, M. Fossorier, and A. Kavcic, "Augmented belief propagation decoding of low-density parity check codes," *IEEE Trans. on Comm.,*, vol. 55, no. 7, pp. 1308–1317, July 2007.

[3] N. Varnica and M. Fossorier, "Improvements in belief-propagation decoding based on averaging information from decoder and correction of clusters of nodes," *IEEE Comm. Letters*, vol. 10, no. 12, pp. 846–848, Dec. 2006.

[4] H. Pishro-Nik and H. Fekri, "Improved decoding algorithms for low-density parity-check codes," in *Int. Conf. Turbo Codes Relat. Top.*, Brest, France, 2003.

[5] C. Measson, A. Montanari, and R. Urbanke, "Maxwell construction: The hidden bridge between iterative and maximum a posteriori decoding," *IEEE Trans. on Inf. Theory,*, vol. 54, no. 12, pp. 5277–5307, Dec 2008.

[6] J. Kang, Q. Huang, S. Lin, and K. Abdel-Ghaffar, "An iterative decoding algorithm with backtracking to lower the error-floors of LDPC codes," *IEEE Trans. on Comm.,*, vol. 59, no. 1, pp. 64–73, Jan. 2011.

[7] W. Ryan and S. Lin, *Channel Codes: Classical and Modern.* Cambridge University Press, 2009.

[8] R. Tanner, "A recursive approach to low complexity codes," *IEEE Trans. on Inf. Theory,*, vol. 27, no. 5, pp. 533–547, Sep 1981.

[9] S. Gounai and T. Ohtsuki, "Decoding algorithms based on oscillation for low-density parity check codes," *IEICE Trans. on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E88-A, no. 8, pp. 2216–2226, August 2005.

[10] M. Zhou and X. Gao, "Improved augmented belief propagation decoding based on oscillation," in *A.-P. Conf. on Comm.*, Oct. 2008, pp. 1–4.

[11] S. Gounai, T. Ohtsuki, and T. Kaneko, "Modified belief propagation decoding algorithm for low-density parity check code based on oscillation," in *IEEE VTC*, vol. 3, May 2006, pp. 1467–1471.

[12] G. Lechner, "Convergence of sum-product algorithm for finite length low-density parity-check codes," in *Winter School on Coding and Inf. Theory*, Ascona, Switzerland, February 2003.

[13] D. Declercq, E. Li, B. Vasic, and S. Planjery, "Approaching maximum likelihood decoding of finite length LDPC codes via FAID diversity," in *IEEE ITW*, Sept 2012, pp. 487–491.