Pivoting Algorithms for Maximum Likelihood Decoding of LDPC Codes over Erasure Channels

Gianluigi Liva and Balazs Matuz
Institute of Communication and Navigation
Deutsches Zentrum fur Luft- und Raumfahrt (DLR)
82234 Wessling, Germany
{Gianluigi.Liva, Balazs.Matuz}@dlr.de

Enrico Paolini and Marco Chiani
DEIS, WiLAB
University of Bologna
via Venezia 52, 47023 Cesena (FC), Italy
{e.paolini, marco.chiani}@unibo.it

***Abstract*—This paper investigates efficient maximum-likelihood (ML) decoding algorithms for low-density parity-check (LDPC) codes over erasure channels. In particular, enhancements to a previously proposed structured Gaussian elimination approach are presented. The improvements are achieved by developing a set of algorithms, here referred to as pivoting algorithms, aiming to limit the average number of reference variables (or pivots) from which the erased symbols can be recovered. Four pivoting algorithms are compared, which exhibit different trade-offs between the complexity of the pivoting phase and the average number of pivots. Numerical results on the performance of LDPC codes under ML erasure decoding complete the analysis, confirming that a near-optimum performance can be obtained with an affordable decoding complexity, up to very high data rates. For example, for one of the presented algorithms, a software implementation has been developed, which is capable to provide data rates above 1.5 Gbps on a commercial computing platform.**

***Index Terms*—Low-density parity-check codes, maximum likelihood decoding, erasure channel, Gaussian elimination.**

I. Introduction

Recently, the design of codes and decoders for erasure channels has attracted an increasing attention [1]–[4]. Among the applications of erasure correcting codes, the recovery of lost packets in wireless communication systems is particularly appealing from a practical viewpoint. A systematic erasure correcting code is used to protect data packets by adding redundant (parity) packets. This redundancy is exploited at the receiver side to recover the lost data packets, provided a sufficient number of packets have been received.

An increasing number of mobile wireless broadcasting/multicasting communication standards are recommending the use of erasure correcting codes as a countermeasure for packet losses due to fading events [5]–[7]. Furthermore, erasure correcting codes are being investigated within the Consultative Committee for Space Data Systems (CCSDS) as an efficient alternative to Automatic Repeat-reQuest (ARQ) protocols for recovering lost packets in deep-space communication links [8]. Codes for packet loss recovery have been proposed also in the context of both distributed storage systems [9] and free-space optical links [10]. All the above-mentioned applications are particularly demanding in terms of performance and data rates, which leads to the need of erasure correcting codes with a near-optimum symbol recovery capability under low-complexity decoding.[1]

Among the different classes of erasure correcting codes, low-density parity-check (LDPC) codes [11] have gained a great importance, thanks to their capacity-approaching performance under efficient iterative (IT) decoding. IT decoding of LDPC codes has been shown to provide an extraordinary error correction capability, over a wide range of communication channels, at a low complexity. Especially for large code lengths, a close-to-capacity behavior can be observed. In the specific case of the binary erasure channel (BEC), LDPC ensembles achieving the channel capacity under IT decoding have been derived, both with an unbounded (e.g. [12]) and with a bounded (e.g. [13]) graphical complexity. However, for code lengths of practical use, the IT performance curve of an LDPC code usually exhibits a non-negligible coding gain loss with respect to that of the same code under maximum likelihood (ML) decoding. This performance gap can be observed both at high erasure probabilities, as a coding gain loss in the waterfall region, and at low erasure probabilities, as a higher error floor [14]. Over the BEC, this is due to the presence of stopping sets [15] not associated with codewords, and thus resolvable by an ML decoder, but not by an IT decoder.

Over the BEC, ML decoding of a binary linear block code consists of solving by matrix inversion the linear equation

$$\mathbf{x}_{\overline{K}}\,\mathbf{H}_{\overline{K}}^{T} = \mathbf{x}_{K}\,\mathbf{H}_{K}^{T} \tag{1}$$

where $\mathbf{x}_{\overline{K}}$ (resp. $\mathbf{x}_K$) denotes the set of erased (resp. correctly received) encoded bits and $\mathbf{H}_{\overline{K}}$ (resp. $\mathbf{H}_K$) the submatrix composed of the corresponding columns of the $(m \times n)$ parity-check matrix $\mathbf{H}$ (where m is the number of parity-check equations and n is the codeword length). Then, ML decoding over the BEC is equivalent to performing a Gaussian elimination (GE) on the binary matrix $\mathbf{H}_{\overline{K}}$. The complexity of ML decoding over the BEC is then $O(n^3)$. Even if ML decoding becomes impractical for long block lengths, for LDPC codes it is possible to take advantage of the parity-check matrix sparseness to reduce the complexity of GE, making it

[1] Throughout the paper, by *symbol* we denote either a bit or a packet of bits. In this paper we will usually assume a bit-oriented perspective, the extension to packets of bits being straightforward.

978-1-4244-4148-8/09/$25.00 ©2009

feasible the use of ML decoding based on the solution of (1) up to codeword lengths even in the order of a few thousands of symbols. An efficient ML decoding algorithm for LDPC codes, exploiting the sparseness of the parity-check matrix, was proposed in [16].

In this paper, we elaborate on the ML decoding algorithm proposed in [16] to show that its complexity can be further reduced. This goal is achieved by adopting smart techniques, here referred to as *pivoting algorithms*, limiting the complexity of the stage of the algorithm in [16] dominating the overall complexity. Numerical results on the performance of LDPC codes under ML erasure decoding complete the analysis, confirming that a near-optimum performance can be obtained with an affordable decoding complexity, up to very high data rates. For example, it is illustrated how fast ML decoders for packet erasure correcting LDPC codes can be built, reaching decoding speeds up to several Gbps via software implementations running on commercial computing platforms.

The paper is organized as follows. Some preliminaries are introduced in Section II. Section III introduces the general framework of ML decoding of LDPC codes over the BEC. Enhanced techniques to reduce the complexity of ML decoding are presented in Section IV. Numerical results are then presented in Section V, to illustrate the gain achievable by adopting the proposed methods. Conclusions follow in Section VI.

II. Variable Nodes, Components and Super Variable Nodes

For any binary matrix $\mathbf{M}$ we say that the row of index i and the column of index j are connected if the (i,j)th entry of $\mathbf{M}$ is equal to '1'. Moreover, we say that two rows of $\mathbf{M}$ are connected when there exists at least one column that is connected to both rows. The weight of a row is defined as the number of '1' entries in the row. Analogously, the weight of a column is defined as the number of '1' entries in the column. We call *accumulated column weight* of a row the sum of the weights of the columns connected with that row.

Any binary matrix $\mathbf{M}$ of size $(m \times n)$ can be represented as a bipartite graph $\mathcal{G} = \{V, C, E\}$ composed of a set V of n variable nodes (VNs), one for each column of $\mathbf{M}$, a set C of m check nodes (CNs), one for each row of $\mathbf{M}$, and a set E of edges. An edge connects a variable node $v_j \in V$ to a check node $c_i \in C$ if and only if $m_{ij} = 1$, where m_{ij} is the (i,j)th element of $\mathbf{M}$. The degree of a VN is equal to the weight of the corresponding column of $\mathbf{M}$. The degree of a CN is equal to the weight of the corresponding row of $\mathbf{M}$. A bipartite graph $\mathcal{G} = \{V, C, E\}$ is defined to be *connected* whenever, for each pair of nodes (v_j, c_i), with $v_j \in V$ and $c_i \in C$, there exists a path in $\mathcal{G}$ from v_j to c_i. An analogous definition shall be adopted for any subgraph of $\mathcal{G}$. The representation of a binary matrix as a bipartite graph was introduced in [17] to represent the parity-check matrix $\mathbf{H}$ of an LDPC code. Within this context, the bipartite graph is sparse and is known as a Tanner graph.

Given a bipartite graph $\mathcal{G} = \{V, C, E\}$, we define subgraph induced by a subset $\tilde{V} \subseteq V$ the bipartite graph composed of $\tilde{V}$, of the subset $\hat{C} \subseteq C$ connected to $\tilde{V}$ and of the edges connecting $\tilde{V}$ to $\hat{C}$. An analogous definition shall be adopted for the subgraph induced by a subset $\tilde{C} \subseteq C$.

Next, we define the concepts of maximal component and of super variable node of a bipartite graph. The former was introduced in [7, Annex E] within the context of ML decoding of Raptor codes for Multimedia Broadcast Multicast Service (MBMS). The latter is introduced in this paper.

Definition 1 (Component of a bipartite graph): In a bipartite graph, a subgraph induced by a set of degree-2 CNs is said to be a component when it is connected. The number of VNs connected to the set of degree-2 CNs is said to be the size of the component.

Definition 2 (Maximal component of a bipartite graph): In a bipartite graph, a component of size q is said to be a maximal component when there exist no components whose size is larger than q.

Definition 3 (Super variable node of a bipartite graph): In a bipartite graph, a subgraph induced by a subset of the CNs is said to be a super variable node (SVN) when it includes at least one VN v for which the following holds: if the value of v is known and the value of all the other VNs in the subgraph is unknown, then the IT decoder run on the subgraph is capable to recover from the erasure pattern. The VN v is said to be a *key node* of the SVN. The number of VNs included in the SVN is said to be the size of the SVN, while the number of edges emanating from the SVN (towards CNs that are not included in the SVN) is said to be the degree of the SVN.

The concept of SVN generalizes that of component in that any component is also a SVN where all VNs are key nodes. Moreover, it is easy to construct simple examples of SVNs which are not components (for instance, because they include CNs of degree larger than 2). In terms of stopping sets, any component of size q does not contain any stopping set of size size smaller than q. On the other hand, a SVN of size q may contain stopping sets of size $q-1$, but at least one subset of size $q-1$ of its VNs does not form a stopping set. In Fig. 1 an example of component (a) and of SVN (b) are depicted.

In a bipartite graph, VNs, components and SVNs are connected through CNs. In some cases, if a CN is connected only to two such structures, merging the two structures leads to either a component or a SVN. An overview of all *admissible merging operations* is depicted in Fig. 2. For example, merging two components each one having a single connection towards the same degree-2 CN produces a component of larger size (case c). As another example, merging a SVN having either a single or multiple connections towards a CN with a component having a single connection towards the same CN produces a SVN of larger size (case g). The cases not reported in Fig. 2 are those for which neither a component nor a SVN is obtained.

If an (n,k) linear block code is used to communicate over a BEC, its codeword symbols are either correctly received or erased. The number of erased symbols in the received

978-1-4244-4148-8/09/$25.00 ©2009
This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE "GLOBECOM" 2009 proceedings.

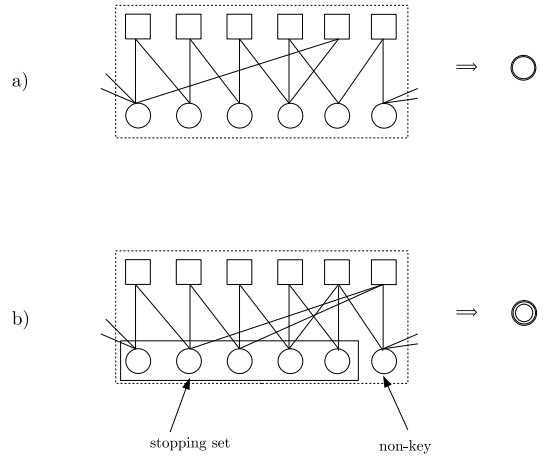


Fig. 1. Example of a component (a) and of a SVN (b). Each VN in the component is a key node. On the other hand, the right-most VN in the SVN is not a key node, while all the other VNs in the SVN are key nodes.

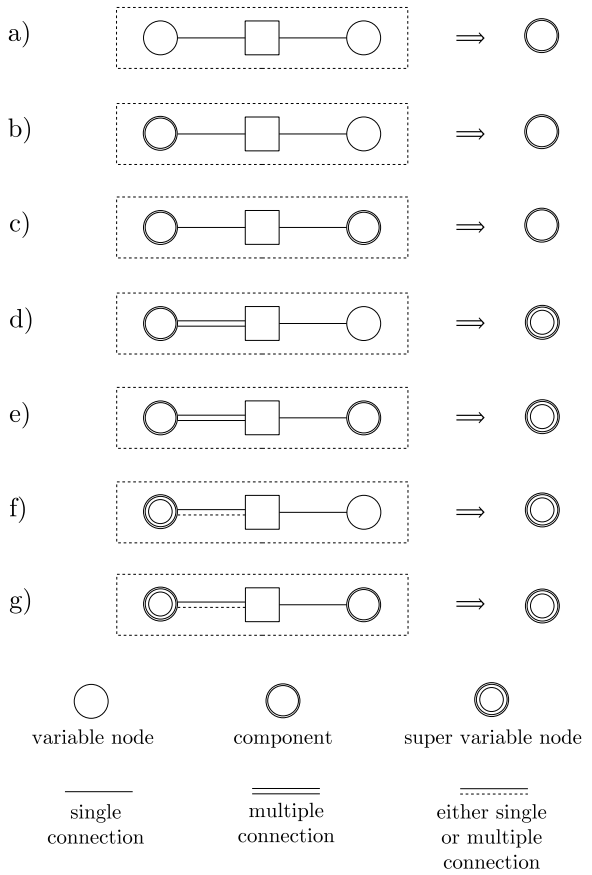


Fig. 2. Summary of admissible node merging operations.

codeword is denoted by e, for $e \in \{0, \ldots, n\}$, and is referred to as the erasure pattern size. The *overhead*, denoted by δ in the following, is defined as the number of correctly received symbols in excess with respect to k, that is $\delta = n - k - e$. The channel erasure probability is denoted by ϵ.

III. Efficient ML decoding of LDPC Codes over the BEC

ML decoding of LDPC codes over the BEC can be practically implemented thanks to a reduced complexity approach [16] which takes its inspiration from a class of structured GE algorithms [18]. (Starting from that, a hybrid iterative-maximum likelihood decoder has been proposed in [14].) It is described in the following.

Reduced complexity ML decoding for LDPC codes over the BEC proceeds as follows. Instead of applying a brute-force GE to solve (1), the following three steps are executed to reduce the decoding complexity without affecting the performance.

Reduced-complexity ML Decoder of LDPC Codes over the BEC:

1. *Triangularization procedure.* $\mathbf{H}_{\overline{K}}$ is transformed into an approximate triangular matrix, as depicted in Fig. 3(a) by row and column permutations only. The obtained matrix is composed of a lower triangular matrix $\mathbf{T}$ and of the three sparse matrices $\mathbf{C}$, $\mathbf{R}_U$, $\mathbf{R}_L$. Some of the columns blocking the triangularization process have been moved to the rightmost part of $\mathbf{H}_{\overline{K}}$ and hence form $\mathbf{R}_U$ and $\mathbf{R}_L$. The α unknowns associated with such columns are referred to as the *reference variables* or as the *pivots*.
2. *Zero matrix procedure.* $\mathbf{T}$ is transformed into an identity matrix by row additions. Moreover, $\mathbf{C}$ is made equal to the zero matrix by row additions, leading to the matrix depicted in Fig. 3(b). Note that, due to the row additions, both $\mathbf{R}_U$ and $\mathbf{R}_L$ usually become dense.
3. *Gaussian elimination procedure.* GE is applied to $\mathbf{R}_L$ to recover the α reference variables. The remaining $e - \alpha$ unknowns are solved by simple substitution.

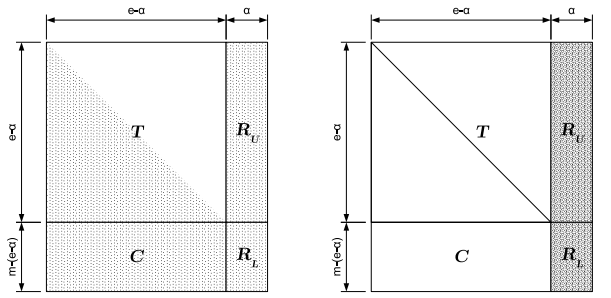
During the triangularization procedure, the elements of the known vector $\mathbf{s}^T \triangleq \mathbf{x}_K \mathbf{H}_K^T$ (see (1)) are permuted according to the row permutations performed on $\mathbf{H}_{\overline{K}}$, leading to $\hat{\mathbf{s}}^T$. Similarly, during the zero matrix procedure, the elements of $\hat{\mathbf{s}}^T$ are summed according to the row additions performed on $\mathbf{H}_{\overline{K}}$.

From a complexity viewpoint, the critical step of this ML decoding algorithm is represented by the Gaussian elimination procedure (step 3) [16]. Although the overall complexity remains $O(n^3)$, the complexity of recovering from a pattern of $e = \chi n$ erasures is reduced by a factor of at least $(\chi n/\alpha)^2$ with respect to a Gaussian elimination performed on $\mathbf{H}_{\overline{K}}$, where α is the final number of pivots.

A decoding failure takes place if the rank of $\mathbf{R}_L$ is smaller than α. Note that the performance of this reduced complexity ML decoder is exactly the same as that achieved by brute-force GE performed on $\mathbf{H}_{\overline{K}}$. In fact, any erasure pattern correctable by GE performed on $\mathbf{H}_{\overline{K}}$ can be corrected by this decoder, and vice versa. The main advantage of this approach relies on its capability to take advantage of the sparseness of the matrix $\mathbf{H}_{\overline{K}}$ to considerably reduce the size of the matrix $\mathbf{R}_L$, on which GE has to be applied.

The size of the matrix $\mathbf{R}_L$ at the end of the triangularization procedure depends on the subroutine adopted to reduce $\mathbf{H}_{\overline{K}}$ in an approximate triangular form. A closer look at $\mathbf{H}_{\overline{K}}$ during

978-1-4244-4148-8/09/$25.00 ©2009
This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE "GLOBECOM" 2009 proceedings.



(a) Structure of $\mathbf{H}_{\overline{K}}$ at the end of the triangularization procedure.

(b) Structure of $\mathbf{H}_{\overline{K}}$ at the end of the zero matrix procedure.

Fig. 3. Efficient Gaussian elimination steps on the $(m \times e)$ matrix $\mathbf{H}_{\overline{K}}$.

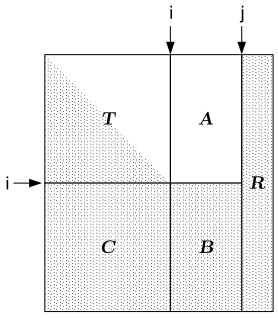


Fig. 4. Structure of $\mathbf{H}_{\overline{K}}$ during the triangularization procedure.

the triangularization procedure shows that it can be divided into the sub-matrices illustrated in Fig. 4. Here, $\mathbf{B}$ is the submatrix that has still to be put in a triangular form and $\mathbf{R}$ is composed of $\mathbf{R}_U$ and $\mathbf{R}_L$. Note that, at the beginning of the procedure, we have $\mathbf{B} = \mathbf{H}_{\overline{K}}$. With reference to Fig. 4, the triangularization procedure (step 1 of the decoding algorithm) may be formalized as follows.

Triangularization procedure:

1. Set $i = 0$ and $j = e - 1$. Set $\mathbf{B} = \mathbf{H}_{\overline{K}}$.
2. Search for a weight-1 row of $\mathbf{B}$. If multiple weight-1 rows exist, select one randomly. If no weight-1 row exists, goto 4.
3. Move the '1' entry of the selected weight-1 row to the top-left position of $\mathbf{B}$ through row and column permutations only. Set $i = i + 1$. If $i = j$ exit. Else goto 1.
4. Choose $\gamma \geq 1$ columns of $\mathbf{B}$ according to a certain *pivoting algorithm*. Move to $\mathbf{R}$ the corresponding column(s) of $\mathbf{H}_{\overline{K}}$. Set $j = j - \gamma$. If $i = j$ exit. Else goto 2.

The pivoting algorithm has a strong impact on the final size of $\mathbf{R}_L$, that is on the number α of pivots, and therefore on the decoder speed. Some pivoting algorithms are described in the next section. Their effectiveness in keeping the dimension of $\mathbf{R}_L$ small is then illustrated through numerical examples in Section V. Among the described pivoting algorithms, three are proposed in this paper, while one (maximal component pivoting) appears in [7, Annex E].

IV. Pivoting Algorithms

If the triangularization procedure gets stuck due to the lack of weight-1 columns in $\mathbf{B}$, $\gamma \geq 1$ columns are selected and moved to $\mathbf{R}$. Good candidates for reference variables are those columns of $\mathbf{H}_{\overline{K}}$ for which we have a high probability to unblock the triangularization procedure and to make it continue for a number of steps as long as possible before it eventually stops again. Several pivoting algorithms are proposed in the following.

A. Maximum Column Weight (MCW) Pivoting Algorithm

It is advisable to select as a pivot a column such that the weight of a high number of rows of $\mathbf{B}$ is decreased by one. In fact, in this way we may obtain a high number of weight-1 rows in $\mathbf{B}$. This can be achieved by selecting a column of $\mathbf{B}$ with the highest weight. Note that in this case we always have $\gamma = 1$.

Algorithm 1 (MCW Pivoting):

a. Calculate the weights of the columns of $\mathbf{B}$.[2]
b. If there exists only one column with maximum weight, select this column.
c. Else select randomly a column of maximum weight.
d. Move to $\mathbf{R}$ the column of $\mathbf{H}_{\overline{K}}$ associated with the selected column of $\mathbf{B}$.

B. Maximum Accumulated Weight (MAW) Pivoting Algorithm

The MCW pivoting algorithm has the advantage to lead to the selection of only one pivot per time ($\gamma = 1$). However, there is no guarantee to unblock the triangularization procedure, so that it may be necessary to it several times before the triangularization can continue. An alternative strategy consists of selecting as pivots columns that are connected to low-weight rows of $\mathbf{B}$. The following pivoting algorithm admits $\gamma \geq 1$, but always guarantees unblocking of the triangularization procedure.

Algorithm 2 (MAW Pivoting):

a. Calculate the weights of the rows of $\mathbf{B}$.
b. If there exists only one row of minimum weight, select this row.
c. Else calculate the accumulated column weight of each minimum weight row. Select randomly a row of minimum weight and maximum accumulated column weight.
d. Move to $\mathbf{R}$ the columns of $\mathbf{H}_{\overline{K}}$ associated with all columns of $\mathbf{B}$ connected to the chosen row, but one (selected randomly).

C. Maximal Component (MC) Pivoting Algorithm

If the triangularization procedure gets stuck, the matrix $\mathbf{B}$ may exhibit several weight-2 rows, and some of these rows may be connected to each other to form a component in the bipartite graph representation of $\mathbf{B}$. Since each VN of a component of size q is a key node, selecting any VN of a

[2]Note that it is not necessary to calculate the weights of the columns of $\mathbf{B}$ at each call to the algorithm, as these values can be tracked during the triangularization procedure. The same observation shall apply to the computation of the weights of the rows of $\mathbf{B}$ in Algorithm 2 and Algorithm 3.

978-1-4244-4148-8/09/$25.00 ©2009
This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE "GLOBECOM" 2009 proceedings.

component as a pivot unblocks the triangularization procedure for at least $q-1$ steps. Therefore it is beneficial to select as a pivot any column of $\mathbf{H}_{\overline{K}}$ associated with a VN belonging to a maximal component in the bipartite graph representation of $\mathbf{B}$.

Algorithm 3 (MC Pivoting):

a. Calculate the weights of the rows of $\mathbf{B}$.
b. If there exist no weight-2 rows then, among the rows of minimum weight in $\mathbf{B}$, choose randomly one row having minimum weight in $\mathbf{H}_{\overline{K}}$.
c. Else find the maximal components in the bipartite graph representation of $\mathbf{B}$. Choose randomly one weight-2 row of $\mathbf{B}$ from a (randomly selected) maximal component.
d. Choose as pivots all columns of $\mathbf{H}_{\overline{K}}$ connected to the selected row of $\mathbf{B}$, but one (selected randomly).

For an exhaustive description of how the maximal components of $\mathbf{B}$ can be identified we refer to [7, Annex E]. The MC pivoting algorithm leads to $\gamma = 1$ when weight-2 rows of $\mathbf{B}$ exist and to $\gamma > 1$ otherwise.

D. SVN Pivoting Algorithm

The algorithm consists of two parts: a growth phase, where several nodes are merged into SVNs and components, and a pivoting phase, where the selection of a pivot is performed. During the growth phase, the bipartite graph associated with $\mathbf{B}$ is modified as follows. In a preliminary stage, all the VNs which are connected by degree-2 CNs are merged, forming components (Fig. 2a). In the resulting graph, some CNs may be connected either to two components or to a component and a VN. In the first case, if the CN is connected to at least one component through a single connection, then the nodes can be merged to form either a component (Fig. 2c) or a SVN (Fig. 2e). Similarly, in the second case the component and the VN can be merged to form either a component (Fig. 2b) or a SVN (Fig. 2d). In a further step, VNs, components and SVNs can be further merged, following the rules on admissible merges depicted in Fig. 2. The algorithm proceeds as follows.

Algorithm 4 (SVN Pivoting):

a. Calculate the degree of each VN, component and SVN in the graph obtained at the end of the growth phase, including in the calculation the multiple connections.
b. Select the node with the largest degree.
c. If the selected node is a VN, move to $\mathbf{R}$ the column of $\mathbf{H}_{\overline{K}}$ associated with it.
d. If the selected node is a component, move to $\mathbf{R}$ the column of $\mathbf{H}_{\overline{K}}$ associated with a randomly selected VN belonging to the component.
e. If the selected node is a SVN, move to $\mathbf{R}$ the column of $\mathbf{H}_{\overline{K}}$ associated with a randomly selected key node belonging to the SVN.

Once the triangularization procedure gets stuck, the above-presented algorithm is applied, starting from the growth phase. The SVN pivoting algorithm always leads to $\gamma = 1$.

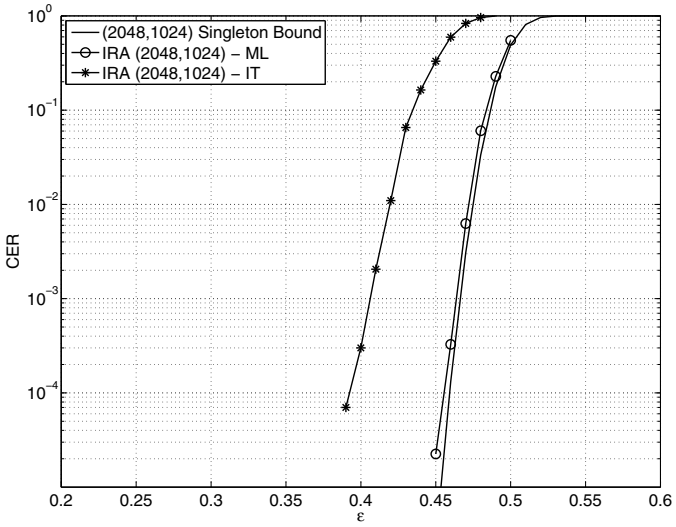


Fig. 5. Performance of the $(2048, 1024)$ IRA code with IT and ML decoding on the BEC.

V. Numerical Results

This section provides a comparison among the pivoting algorithms introduced in Section IV. In particular, we investigate through simulation the impact of the algorithms on the average number of pivots. We then provide an estimation on the achievable throughputs for an actual software-based implementation of the ML decoder. For our investigations, we consider a $(2048, 1024)$ systematic Irregular Repeat-Accumulate (IRA) code [19] with degree distributions

$$\begin{aligned}\lambda(x) = {} & 0.222x + 0.216x^2 + 0.031x^6 + 0.192x^8 + \\ & + 0.047x^{17} + 0.075x^{18} + 0.217x^{53}, \\ \rho(x) = {} & x^8,\end{aligned}$$

whose parity-check matrix has been constructed according to the progressive edge-growth (PEG) algorithm [20]. For sake of completeness, the performance of the code under IT and ML decoding in terms of codeword error rate (CER) vs. erasure probability (ϵ) is reported in Fig. 5. The Singleton bound for (2048,1024) codes is depicted as well.

Simulation results on the average number of pivots ($\overline{\alpha}$) vs. overhead are depicted in Fig. 6. Each value of $\overline{\alpha}$ is obtained by averaging over 100 runs. As a reference, the curve obtained for a random pivoting is provided. A first remark relates to the fact that even the MCW algorithm (which is the simplest among those presented in this paper) leads to a remarkable reduction in the average number of pivots (by a factor of about 3) with respect to the random pivoting case. Further, but more limited, gains can be achieved by adopting smarter algorithms. The most efficient algorithm results to be the SVN pivoting one, which saves on average 4 to 5 pivots with respect to MCW and almost 1 pivot with respect to the MC algorithm.

A final test has been carried out on a software implementation of the erasure decoder based on the ML approach discussed in this paper. The MCW algorithm has been used for the selection of pivots and the net data rate has been measured

978-1-4244-4148-8/09/$25.00 ©2009
This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE "GLOBECOM" 2009 proceedings.

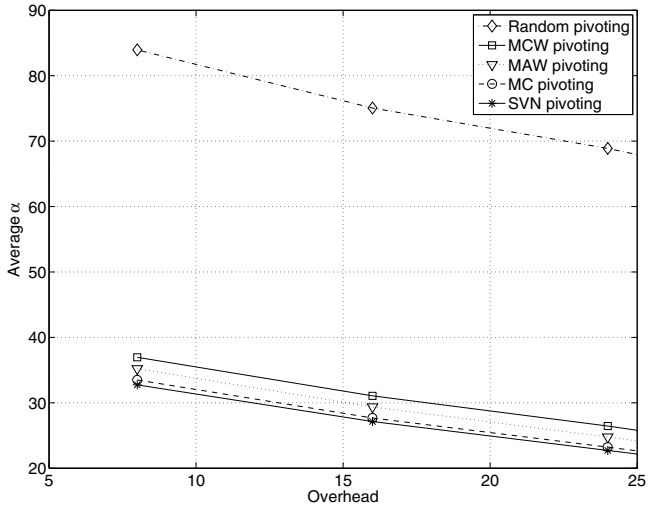


Fig. 6. Comparison among different pivoting algorithms in terms of average number of pivots ($\overline{\alpha}$) vs. overhead.

at the decoder output, where by net data rate we mean the number of processed information bits per second. The decoder has been run on a commercial PC with a 3 GHz Intel CPU and 8 Gbytes of RAM. The symbol (packet) size was set to 1024 bytes. The ML decoder achieves a minimum data rate of about 1.5 Gbps for $\delta = 8$ symbols of overhead, while the IT decoder provides data rates larger than 4.5 Gbps. For erasure patters of moderate size (e.g., overheads in the order of 150 symbols) both the IT and the ML decoder achieved net data rates in the order of 5 Gbps.

This decoder implementation was actually tested in field trials within the ESA J-ORTIGIA project, where it has been integrated within the MPE-iFEC protocol for packet loss recovery in a DVB-SH-like receiver prototype [6]. The tests where performed by broadcasting the encoded stream in S-band through the ETS-VIII geostationary satellite. On the receiver side, after physical layer decoding, the MPE-iFEC stream was processed in real-time by an ultra-portable PC (equipped with an 800 MHz CPU) on which the ML erasure decoder was running. The tests confirmed the validity of the approach both in terms of decoding speed (it was actually possible to decode several parallel streams simultaneously) and performance (a near-optimum performance has been obtained).

VI. Concluding Remarks

In this paper, pivoting algorithms for efficient ML decoding of LDPC codes over erasure channels have been proposed and their performance has been investigated through numerical simulation. Out of the four described pivoting algorithms, two are very simple (MCW and MAW), while two are more complex (MC, which appeared in [7], and SVN). Our simulation results suggest that the two simple pivoting algorithms achieve a performance (in terms of average number of pivots) quite close to that of more complex techniques. Moreover, it has been illustrated how the SVN pivoting algorithm can outperform the MC pivoting one. For the MCW algorithm, a software implementation has been developed, which is capable of providing data rates above 1.5 Gbps on a commercial computing platform. This confirms that a near-optimum performance can be obtained with an affordable decoding complexity, up to very high data rates.

Acknowledgment

The work was supported in part by the EC under Seventh FP grant agreement ICT OPTIMIX nINFSO-ICT-214625 and by the EC-IST SatNEx-II project (IST-27393).

References

[1] J. Byers, M. Luby, M. Mitzenmacher, and A. Rege, "A digital fountain approach to reliable distribution of bulk data," *SIGCOMM Comput. Commun. Rev.*, vol. 28, no. 4, pp. 56–67, Oct. 1998.

[2] M. Luby, M. Mitzenmacher, M. Shokrollahi, and D. Spielman, "Efficient erasure correcting codes," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 569–584, Feb. 2001.

[3] M. Luby, "LT codes," in *Proc. of the 43rd IEEE Symp. on Foundations of Computer Science*, Vancouver, Canada, Nov. 2002, pp. 271–282.

[4] M. Shokrollahi, "Raptor codes," *IEEE Trans. Inf. Theory*, vol. 52, no. 6, pp. 2551–2567, June 2006.

[5] "Transmission System for Handheld Terminals (DVB-H)," Digital Video Broadcasting (DVB)," Blue Book, 2004.

[6] "Framing structure, channel coding and modulation for Satellite Services to Handheld devices (SH) below 3GHz," Digital Video Broadcasting (DVB)," Blue Book, 2007.

[7] 3GPP TS 26.346 V8.0.0, "Technical specification group services and system aspects; multimedia broadcast/multicast service; protocols and codecs," Sept. 2008.

[8] G. P. Calzolari, M. Chiani, F. Chiaraluce, R. Garello, and E. Paolini, "Channel coding for future space missions: New requirements and trends," *Proc. IEEE*, vol. 95, no. 11, pp. 2157–2170, Nov. 2007.

[9] A. Dimakis, V. Prabhakaran, and K. Ramchandran, "Decentralized erasure codes for distributed networked storage," *IEEE Trans. Inf. Theory*, vol. 52, no. 6, pp. 2809– 2816, June 2006.

[10] G. Liva, T. De Cola, D. Giggenbach, N. Perlot, E. Paolini, M. Chiani, G. P. Calzolari "Packet Loss Recovery via Maximum-Likelihood LDPC Decoding," presented at the CCSDS Fall Meeting, October 2008, Berlin.

[11] R. G. Gallager, *Low-Density Parity-Check Codes*. Cambridge, MA: M.I.T. Press, 1963.

[12] P. Oswald and M. A. Shokrollahi, "Capacity-achieving sequences for the erasure channel," *IEEE Trans. Inf. Theory*, vol. 48, no. 12, pp. 364–373, Dec. 2002.

[13] H. Pfister and I. Sason, "Accumulate-repeat-accumulate codes: Capacity-achieving ensembles of systematic codes for the erasure channel with bounded complexity," *IEEE Trans. Inf. Theory*, vol. 53, no. 6, pp. 2088–2115, June 2007.

[14] E. Paolini, G. Liva, B. Matuz, and M. Chiani, "Generalized IRA erasure correcting codes for hybrid Iterative/Maximum Likelihood decoding," *IEEE Commun. Lett.*, vol. 12, no. 6, pp. 450–452, June 2008.

[15] C. Di, D. Proietti, I. E. Telatar, T. J. Richardson, 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.

[16] D. Burshtein and G. Miller, "An efficient maximum likelihood decoding of LDPC codes over the binary erasure channel," *IEEE Trans. Inf. Theory*, vol. 50, no. 11, pp. 2837–2844, Nov. 2004.

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

[18] B. A. LaMacchia and A. M. Odlyzko, "Solving Large Sparse Linear Systems over Finite Fields," *Lecture Notes in Computer Science*, vol. 537, pp. 109–133, 1991.

[19] H. Jin, A. Khandekar, and R. McEliece, "Irregular repeat-accumulate codes," in *Proc. of the 2nd Int. Symp. on Turbo codes & Related Topics*, Brest, France, Sept. 2000, pp. 1–8.

[20] X. Hu, E. Eleftheriou, and D. Arnold, "Regular and irregular progressive edge-growth Tanner graphs," *IEEE Trans. Inf. Theory*, vol. 51, no. 1, pp. 386–398, Jan. 2005.

978-1-4244-4148-8/09/$25.00 ©2009