A New Upper Bound on the Minimum Distance of Turbo Codes

Alberto Perotti and Sergio Benedetto
Center for Multimedia Radio Communications
Politecnico di Torino – Torino, Italy
Email: {alberto.perotti, sergio.benedetto}@polito.it
Telephone: +39 011 564–4193
Fax: +39 011 564–4099

***Abstract*— In this paper, a new upper bound on the minimum distance of turbo codes is derived. The new bound is obtained by construction of an undirected graph which reflects the characteristics of the constituent codes and the interleaver. The resulting expression shows that the minimum distance of a turbo code grows approximately with the base-3 logarithm of the information word length. The new bound is easy to compute, applies to rate k/n constituent encoders, and often improves over existing results.**

I. Introduction

Turbo-like codes, in their parallel [1] and serial [2] versions, are widely known to be very powerful error correcting codes. Their amazing performance are based on a code construction that privileges very low error event multiplicities, rather than large distances. For this reason, they tend to exhibit an "error floor" that limits their coding gain at very low bit error probabilities [3].

The minimum distance of a turbo-like code is largely determined by the interleaver. In order to avoid the occurrence of low-weight codewords, several methods have been proposed for the design of permutations [4] [5], and some of them are based on graph theory [6]. Moreover, as shown in [7] and [8], the logarithmic dependence on the information word length of the minimum distance is due to the presence of the interleaver.

While average bounds on the maximum-likelihood performance of turbo-like schemes are known [9] [2], based on the concept of the so-called "uniform interleaver", the evaluation of the minimum distance of a scheme using an actual interleaver is a computationally intensive task.

Upper bounds on the minimum distance of parallel concatenated scheme based on known constituent codes as functions of the interleaver size have been proposed in [7] [10] [11] [12]. Among these, the bound in [7] is based on graph theory, and it results in a logarithmic expression. The time required to evaluate the bound is $O(K^2)$, where K is the interleaver length, therefore it becomes impractical for large values of K. For this reason, in the same article a simpler bound is proposed whose complexity is independent of K. In [13], a similar result is obtained which confirms the logarithmic growth of the minimum distance in a two-branch turbo code to be due to the presence of the interleaver.

In this paper, a new upper bound on the minimum distance of parallel concatenated convolutional codes with given information word length and constituent encoder memory is derived. This result is based on the construction of an undirected graph whose characteristics are determined by the constituent encoders and interleaver. The graph construction is similar to that proposed in [13] for the parallel concatenation of accumulators, although our construction also adapts to constituent encoders with larger memory. A correspondence between the set of cycles in the graph and a subset of the turbo codewords is established, and the minimum distance is estimated through the length of the shortest cycle. It is shown that the minimum distance of turbo codes grows approximately with the base-3 logarithm of the information word length. The new bound is fairly simple to compute, and proves to be tighter than the previously known results [7] [10] [11] for a wide range of parameter values.

II. Reference coding scheme

The main characteristics of the code, such as the information word length K, the memory of the constituent encoders ν and the permutation performed by the interleaver, determine the minimum distance of the concatenated code.

The reference coding scheme (see Fig. 1) consists of two equal recursive, convolutional systematic encoders (see Fig. 2 for an example of a 4-state encoder) concatenated in parallel through an interleaver. The information bit sequence $\mathbf{u} = \{u_0, \ldots, u_{K-1}\}$ is sent to the first encoder, which generates the systematic bit sequence $\mathbf{u}$ and the coded bit sequence $\mathbf{c}_1$. The sequence $\mathbf{u}$ is interleaved according to a permutation $\mathbf{\Pi} : i \rightarrow \pi(i) = \pi_i$, $i = 0, \ldots, K-1$ and sent to the second constituent encoder, where the coded bit sequence $\mathbf{c}_2$ is generated. The output of each constituent encoder is punctured to obtain the desired coding rate.

Following the characterization carried out in [7], the constituent encoders are described by the two parameters α and β, which express the Hamming weight of the constituent code sequence $\mathbf{c}_i$ when the information sequence consists of two 1 separated by $kp-1$ zeroes. Here, $p = 2^\nu - 1$ is the *encoder period*. This sequence causes a constituent encoder to leave the all-zero state, and return to it after kp trellis steps:

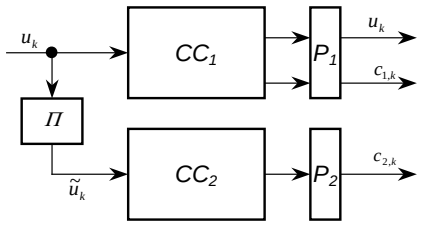


Fig. 1. Reference coding scheme.

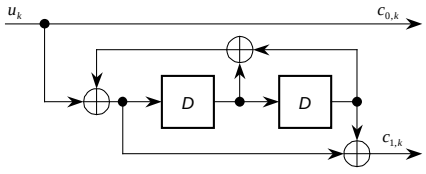


Fig. 2. A 4-state systematic, recursive convolutional encoder.

$$\mathbf{u} = D^i + D^{i+kp}$$

The resulting output Hamming weight is:

$$w_H(\mathbf{c}) \leq \alpha k + \beta \leq kp + 1 \tag{1}$$

where α and β depend on the encoder connections and puncturing pattern.

Here, we will consider periodic puncturing patterns. The systematic bit sequence is not punctured, while the code sequences are punctured to obtain the desired rate. In general, if $\mathbf{p}_i$ is the puncturing pattern applied to the output of CC_i, for a turbo code of rate $k/(k+1)$ we have:

$$\mathbf{p}_1 = 1 + D^{2k} + D^{4k} + \ldots \tag{2}$$
$$\mathbf{p}_2 = D^k + D^{3k} + D^{5k} + \ldots \tag{3}$$
$$\tag{4}$$

where a 1 in $\mathbf{p}_i$ means that the corresponding bit in $\mathbf{c}_i$ is selected, otherwise it is eliminated.

Table I (taken from [10] and here extended) shows a set of example parameter values that will be used to compare our results with [7]. The encoder connections are in octal form. The numerator corresponds to the feedforward connections and the denominator corresponds to the feedback connections.

ν	Connections	p	R			
			1/3	1/2	2/3	7/8
2	$5_8/7_8$	3	2, 2	1, 2	-	-
3	$17_8/13_8$	7	4, 2	2, 2	1, 2	-
4	$35_8/23_8$	15	8, 2	4, 2	2, 2	1, 2

TABLE I

PARAMETERS α, β OF THE CONSIDERED ENCODERS.

Given the set of information words $\mathcal{U}$, and the set of permutations of length K, Π_K, we perform the derivation in the following way:

$$d_m \leq \max_{\Pi_K} \min_{\mathbf{u} \in \mathcal{U}'} w_H(CC_1(\mathbf{u})) + w_H(CC_2(\tilde{\mathbf{u}})) \tag{5}$$

where $\mathcal{U}' \subseteq \mathcal{U}$, and d_m is the minimum distance of the turbo code. $\mathcal{U}'$, contains information words with even Hamming weight.

III. GRAPH NOTATIONS AND PRELIMINARY RESULTS

A graph G is defined by its set of vertices V and set of edges E. An edge can be considered as the association between two vertices $v_i \in V$ and $v_j \in V$, and is usually indicated using the following notation:

$$e = (v_i, v_j)$$

We will consider here only *undirected* edges.

The *degree* of a vertex $v \in V$ is the number of edges incident in v, and is indicated with $d(v)$.

A graph G is called *r-regular* when its vertices have all degree r:

$$\forall v \in V : d(v) = k.$$

3-regular graphs are called *cubic*.

A *cycle* is a subgraph $C = (V', E')$ of $G = (V, E)$, where $V' \subseteq V$ and $E' \subseteq E$, and the following properties are satisfied: (1) every vertex in V' has 2 incident edges, and (2) C is connected, i.e., a path exists between any two vertices of C. The *length* of a cycle C is defined as:

$$l(C) = |V'| = |E'|.$$

The *girth* of G is the length of its shortest cycle:

$$g(G) = \min_{C \in \mathcal{C}}[l(C)] \tag{6}$$

where $\mathcal{C}$ is the collection of all the cycles of G.

An upper bound on the girth of r-regular graphs is obtained following the derivation of the *naive bound* in [17]:

$$g(G) \leq 2\log_3(|V| + 1) - 1 \tag{7}$$

where $|V|$ is the number of vertices of G.

IV. EXTENDED PERMUTATION GRAPHS

The construction of a joint model including the characteristics of the constituent encoders and the interleaver permits to analyze the concatenated encoder as a whole. One way to construct such a model is to consider the permutation graph [14] associated to the interleaver and modify it according to the constituent encoders characteristics.

We will consider *tail-biting* [15] as the trellis termination technique, which consists of forcing each constituent encoder to start and end the block encoding in the same state. The trellis can be considered *circular* and this results in a highly regular extended permutation graph $G_{TB} = (V, E)$, which is defined on the following set of vertices:

$$V = V_1 \cup V_2$$

where

$$\begin{aligned} V_1 &= \{v_{I,j}, j = 0, \ldots, K-1\} \\ V_2 &= \{v_{O,j}, j = 0, \ldots, K-1\} \end{aligned}$$

V_1 (resp. V_2) corresponds to the K bits at the interleaver input (resp. output). The set of edges E is defined as follows:

$$E = E_1 \cup E_\Pi \cup E_2$$

where

$$\begin{aligned} E_1 = &\quad \{(v_{I,j}, v_{I,((j+1) \mod K)}),\ j = 0, \ldots, K-1\} \\ E_\Pi = &\quad \{(v_{I,j}, v_{O,\pi_j}),\ j = 0, \ldots, K-1\} \\ E_2 = &\quad \{(v_{O,j}, v_{O,((j+1) \mod K)}),\ j = 0, \ldots, K-1\} \end{aligned}$$

where E_1 (resp. E_2) contains the edges corresponding to CC_1 error events (resp. CC_2 error events), and E_Π contains the edges corresponding to the interleaver.

In Fig. 3 an example of extended permutation graph for memory 1 constituent encoders is shown.

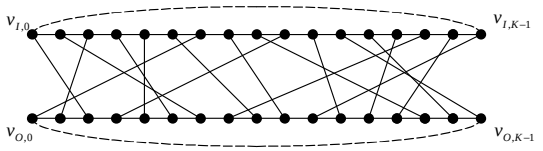


Fig. 3. An example of permutation graph for memory $\nu = 1$ constituent encoders.

The resulting graph is *cubic*.

Two key considerations allow to exploit the characteristics of the extended permutation graph for our derivation:

1) If the constituent encoders are accumulators, i.e., 2-state recursive convolutional encoders, almost all the cycles in the permutation graph can be associated to codewords[1].
2) The length of a cycle is related to the Hamming weight of the associated codeword.

In fact, if $C \subseteq G_{TB}$ is a cycle, we can associate to it an information word $\mathbf{u}$ in the following way:

$$u_k = \begin{cases} 1 & \Leftrightarrow \quad (v_{I,k}, v_{O,\pi_k}) \in C \\ 0 & \Leftrightarrow \quad (v_{I,k}, v_{O,\pi_k}) \notin C \end{cases} \tag{8}$$

i.e., every edge in $C \cap E_\Pi$ corresponds to a 1 in the codeword. Clearly, since $|C \cap E_\Pi|$ is even[2], only information words with *even* Hamming weight are considered. This defines the set $\mathcal{U}'$ of (5).

About the second consideration, the following statements can be formulated:

1) Every vertex $v \in C$ can be associated to a non-zero trellis transition in the corresponding constituent encoder (see Fig. 4).
2) Every path consisting of k consecutive edge in $C \bigcap E_1$ (resp. $C \bigcap E_2$) corresponds to an error event of length $1 + kp$ trellis steps in CC_1 (resp. CC_2).

The two considerations lead to slightly different expressions for the upper bound. However, their values are very close each other. In the following, we will consider the second statement, since it allows an easier extension to the case $\nu > 1$.

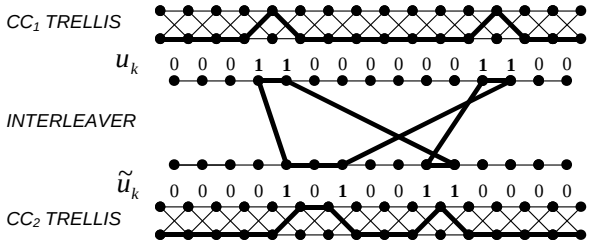


Fig. 4. Correspondence between codewords and cycles.

A. Extension to memory ν constituent encoders

Consider an information word $\mathbf{u}$. If $u_i = 1$, the first constituent encoder leaves the *zero* state at step i. If $u_{i+kp} = 1$, $k \in \mathbb{N}\backslash\{0\}$, the constituent encoder returns to the all-zero state, causing the error event to terminate. This behavior is modelled by connecting vertices $v_{I,i}$ (resp. $v_{O,i}$) and $v_{I,((i+p) \mod K)}$ (resp. $v_{O,((i+p) \mod K)}$) with an edge. The extended permutation graph G_{TB} for period p constituent encoders is defined by its set of edges

$$E = E_1 \cup E_\Pi \cup E_2$$

where

$$\begin{aligned} E_1 = &\quad \{(v_{I,i}, v_{I,(i+p) \mod K}),\ i = 0, \ldots, K-1\} \\ E_\Pi = &\quad \{(v_{I,i}, v_{O,\pi_i}),\ i = 0, \ldots, K-1\} \\ E_2 = &\quad \{(v_{O,i}, v_{O,(i+p) \mod K}),\ i = 0, \ldots, K-1\} \end{aligned}$$

The extended permutation graph for memory $\nu = 3$ constituent encoders is shown in Fig. 5. This graph is cubic, and its girth can be easily bounded from above.

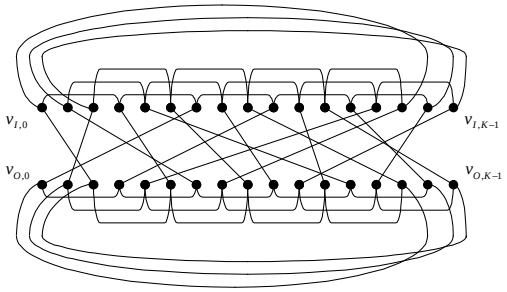


Fig. 5. Extended permutation graph for tail-biting terminated turbo codes.

Similarly to the case $\nu = 1$, cycles correspond to codewords, and their length can be used to estimate the weight of the associated codeword. However, before performing the

[1]We will see later that some cycles do not correspond to codewords. However, under certain constraints, this will not invalidate our result.

[2]This property follows from the bipartite nature of the set of vertices V.

estimation, an improvement can be carried out: given a cycle C, the edges in $C \cap C_\Pi$ do not correspond to error events. A new graph G'_{TB} is obtained from G_{TB} by contracting the edges in E_Π. As shown in Fig. 6, the contraction of an edge incident in two degree-3 vertices results in a new degree-4 vertex.

The resulting graph $G'_{TB} = (V', E')$ is 4-regular and its number of vertices is $|V| = K$. The main advantage is that *all* the edges in G'_{TB} correspond to error events.

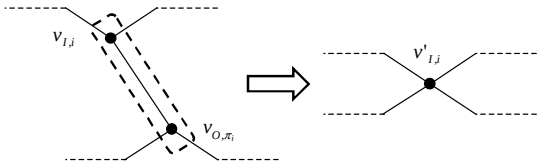


Fig. 6. Contraction of an edge.

Applying (7) with $r = 4$, we obtain:

$$g(G'_{TB}) \leq g_{ub}(G'_{TB}) \leq 2\log_3(K+1) - 1. \tag{9}$$

Eq. (9) provides a preliminary insight on the asymptotical behavior of the minimum distance turbo codes:

$$d_m \leq w \log_3(K)$$

where the w coefficient corresponds to the Hamming weight of the error events.

V. The upper bound

We start from the following consideration: every edge sequence in $G'_{TB} \cap E_1$ and $G'_{TB} \cap E_2$ corresponds to an error event of length $1 + kp$ trellis sections. Using (1), we write the output weight corresponding to an error event of length $1 + kp$ for a *systematic* constituent encoder:

$$w_H(\mathbf{u}) + w_H(\mathbf{c}_1) \leq 2 + k\alpha + \beta$$

Hence, we can define the *maximum equivalent weight* of an error event of length $p + 1$ as

$$w_{e,\max} \triangleq \max_{k>0} \left\lceil \frac{2 + k\alpha + \beta}{k} \right\rceil. \tag{10}$$

which is clearly obtained for $k = 1$:

$$w_{e,\max} = 2 + \alpha + \beta. \tag{11}$$

Putting together (9) and (11), we obtain the following result:

$$d_m \leq w_{e,\max}\, g_{ub}(G'_{TB})$$

or:

$$d_m \leq (\alpha + \beta + 2)\,[2\log_3(K+1) - 1] \tag{12}$$

A slight improvement can be obtained by considering that at least one edge corresponds to CC_2, where the contribution to the output weight of the systematic bits is not present:

$$d_m \leq \alpha + \beta + w_{e,\max}\,(g_{ub}(G'_{TB}) - 1)$$

or, including (11):

$$d_m \leq \alpha + \beta + (\alpha + \beta + 2)\,[2\log_3(K+1) - 2]. \tag{13}$$

This result depends only on few relevant, and easily derived parameters of the turbo code. Moreover, its computation is of very low complexity. However, a simpler expression can be used when the parameters α and β of the constituent encoders are not available. This simpler expression derives from the second part of (1):

$$d_m \leq p + 1 + (p + 3)\,[2\log_3(K+1) - 2]. \tag{14}$$

This result only requires the knowledge of ν, the memory of the constituent encoders.

A. Singular cycles and their associated codewords

Some types of cycles in G_{TB} and G'_{TB} deserve a further analysis: for example, two edges may be partially overlapping (see Fig. 7). The information word associated to this pair of edges corresponds to the modulo-2 sum of two weight-2 information sequences:

$$\mathbf{u} = \mathbf{u}_1 + \mathbf{u}_2 = (D^i + D^k) + (D^j + D^l)$$

and, as long as the constituent codes are linear, the resulting code sequence is the modulo-2 sum of the two error events:

$$\mathbf{c}_1 = CC_1(\mathbf{u}_1) + CC_1(\mathbf{u}_2).$$

This results in a longer error event starting in position i and terminating in position l. In this case, the codeword weight is overestimated, because, in the overlapping region, trellis transition are counted multiple times. This situation results in a possibly looser, but still valid, upper bound.

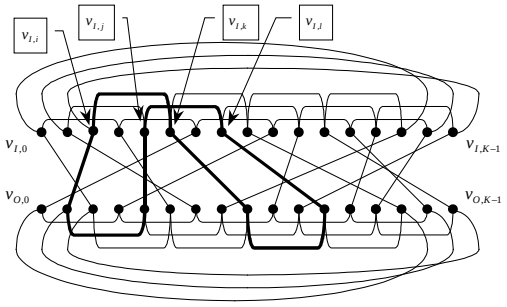


Fig. 7. An example of cycle with partially overlapping error events.

Another class of cycles in G_{TB} has to be considered: cycles containing no edges in E_Π. Since these cycles do not correspond to codewords (or correspond to information words with zero Hamming weight), they could invalidate our result. However, their length is at least $\frac{K}{p}$ (the minimum length is obtained when K is an integer multiple of p), and, under the following constraint:

$$\frac{K}{p} \geq 2\log_3(K+1) - 1 \geq g(G_{TB}) \tag{15}$$

none of these cycles can be the shortest in G_{TB}.

Tab. II shows the minimum values of K for which the upper bound for tail-biting codes is valid for different values of ν.

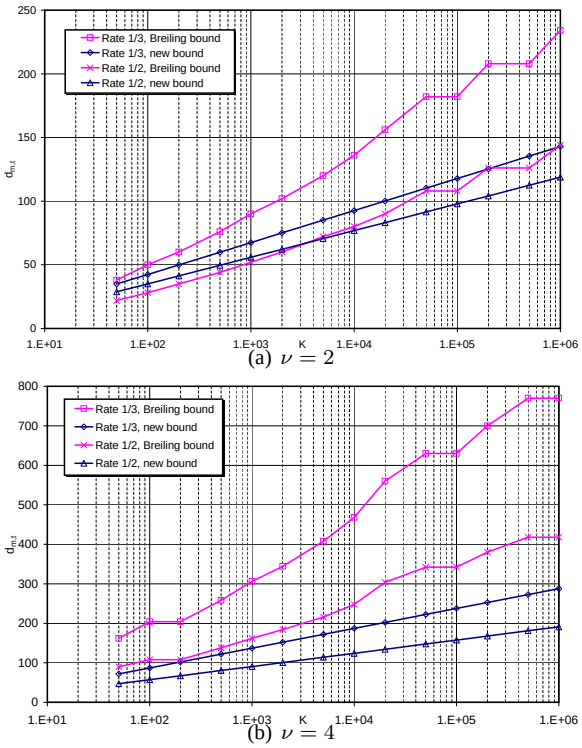
ν	$K_{\min}$
2	11
3	41
4	115

TABLE II

MINIMUM VALUES OF K FOR WHICH THE UPPER BOUND FOR TAIL-BITING CODES IS VALID.

VI. COMPARISON WITH THE PREVIOUSLY KNOWN BOUNDS

This section shows the obtained results for tail-biting turbo codes compared to the best previously known upper bound [7]. Fig. 8 show the new upper bound (13) for independently terminated codes compared to the logarithmic upper bound in [7].



(a) $\nu = 2$

(b) $\nu = 4$

Fig. 8. Upper bounds for rate $^1/_3$ and rate $^1/_2$ codes.

The new bound is significantly tighter than the logarithmic bound in [7] for a wide range of block lengths and encoder memory values, except for memory $\nu = 2$, rate $1/2$ codes. In fact, as explained in Sec. VII, the result in [7] allows to evaluate the exact weight of each considered information word, while the graph defined here only provides an upper bound. Therefore, although our bound remains asymptotically best, the former provides a tighter bound for heavily-punctured codes in the considered range of block lengths.

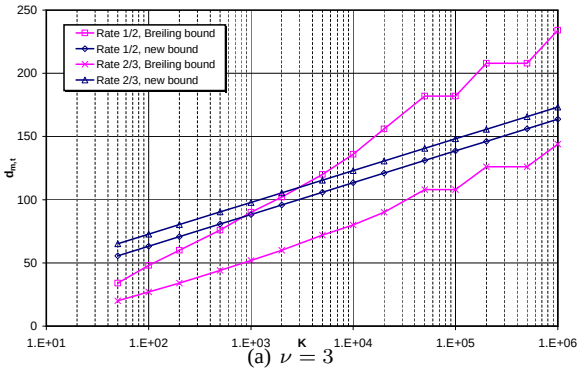
VII. HIGH-RATE TURBO CODES

In order to obtain high rates ($R > 1/2$), the following techniques are usually applied:

1) Puncturing of the coded bit sequence at the output of the constituent encoders (usually performed when the constituent code rate is $1/n$)
2) Use of rate k/n, $k > 1$ constituent encoders

The two techniques can be applied jointly as well; however, we will analyze them separately.

Puncturing consists in eliminating some bits in the code sequence at the output of the constituent encoders, and is usually performed applying a periodic puncturing pattern. The elimination of coded bits results in a reduced value of parameters α and β (see Tab. I). Moreover, due to the characteristics of the iterative decoding algorithm used to decode turbo codes, systematic bits are seldom punctured. This results in a larger number of eliminated coded bits rather than information bits. Fig. 9 shows the obtained results for high-rate turbo codes.



(a) $\nu = 3$

Fig. 9. Upper bounds for high-rate turbo codes ($R > {}^1/_2$).

When rate k/n constituent codes are considered, the information sequence consists of a sequence of K/k (K a multiple of k) symbols.

It is easy to prove that, for linear recursive convolutional codes, two *equal* information symbols whose distance is an integer multiple of p in the information word result in a short error event. Indeed, this is true also when the Hamming weight of the considered information symbol is 1. Therefore, an information word in the form:

$$u(D) = D^i + D^{i+jkp}$$

results in an error event of length $jkp+1$ trellis steps. Hence a modified permutation graph can be constructed even in this case.

When comparing a punctured convolutional encoder with a k/n convolutional encoder with the same rate (puncturing included), the latter possibly results in larger α and β parameters (equal to the parameters for the non punctured

version). Hence, the turbo codes resulting form the parallel concatenation possibly allows a larger minimum distance.

The resulting upper bound for the second code is larger than the upper bound for the first code, as shown in Fig. 10.

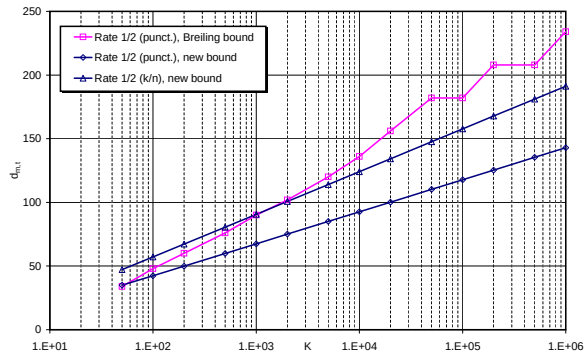


Fig. 10. Upper bounds for a rate $^1/_2$ 8-state turbo code obtained by puncturing alternatively the code bits compared with the parallel concatenation of two systematic rate $()^2/_3$ constituent encoders.

A. Symbol interleaving

Symbol interleaving consists of performing a permutation on the set of symbols instead of bits. Indeed, a permutation

$$\Pi : i \to \pi_i, \quad i \in \left\{0, \ldots, \frac{K}{k_0} - 1\right\}$$

(K an integer multiple of k_0) operating on k_0-bit symbols is equivalent to a permutation $\Pi' : i \to \pi'_i, \quad i \in \{0, \ldots, K-1\}$ having the following property:

$$(i \mod k_0) = (\pi'_i \mod k_0)$$

The resulting extended permutation graph G'_{TB} can be partitioned in k_0 disjoint subgraphs containing K/k_0 vertices each. Its girth can be upper bounded choosing at random one of the k_0 subgraphs and applying (7) with $r = 4$ and $n = K/k_0$. The resulting upper bound is:

$$d_{m,SI} < \alpha + \beta + (\alpha + \beta + 2)\left[2\log_3\left(\frac{K}{k_0} + 1\right) - 2\right] \quad (16)$$

Since usually $K/k_0 \gg 1$, the $+1$ term in the argument of the logarithm can be neglected.

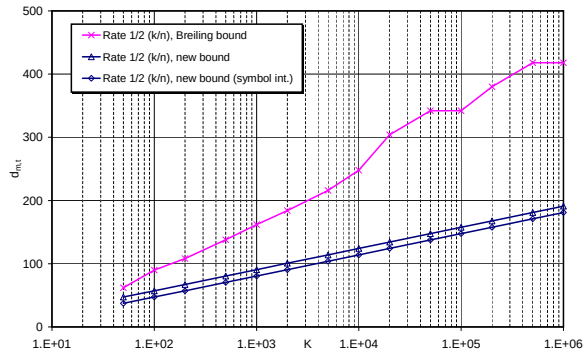


Fig. 11. Effect of symbol interleaving on a rate $1/3$ tail-biting turbo code.

VIII. Conclusions

We have shown in the paper that a connection between parallel concatenated convolutional codes and the topological properties of certain associated graphs can be established. The connection has been used to obtain an upper bound on the minimum distance of parallel concatenated convolutional codes. The proposed method has been extended to rate k/n constituent codes with bit- and symbol-interleaving. The new bound is easy to compute, and improve over known bounds for a wide range of constituent codes parameters.

Acknowledgments

The authors would like to thank Dr.-Ing. Marco Breiling for his valuable support.

References

[1] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near shannon limit error-correcting coding and decoding: turbo codes," in *Proceedings of the International Conference on Communications*, vol. 2, Geneva, Switzerland, 1993, pp. 1064–1070.

[2] S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara, "Serial concatenation of interleaved codes: Performance analysis, design and iterative decoding," *IEEE Transactions on Information Theory*, vol. 44, no. 3, pp. 909–926, May 1998.

[3] R. Garello, P. Pierleoni, and S. Benedetto, "Computing the free distance of turbo codes and serially concatenated codes with interleavers: Algorithms and applications," *IEEE Journal on Selected Areas in Communications*, vol. 19, no. 5, pp. 800–812, May 2001.

[4] S. Dolinar and D. Divsalar, "Weight distributions for turbo codes using random and nonrandom permutations," *TDA Progress Report*, vol. 42-122, pp. 56–65, April-June 1995.

[5] C. Fragouli and R. D. Wesel, "Turbo-encoder design for symbol-interleaved parallel concatenated trellis-coded modulation," *IEEE Transactions on Communications*, vol. 49, no. 3, pp. 425–435, March 2001.

[6] P. O. Vontobel, "On the construction of turbo code interleavers based on graphs with large girth," in *Proceedings of the International Conference on Communications*, New York, USA, 2002, pp. 1408–1412.

[7] M. Breiling and J. B. Huber, "A logarithmic upper bound on the minimum distance of turbo codes." [Online]. Available: http://www.lnt.de/~breiling/Research/Publications/bre01b.pdf

[8] D. Truhachev, M. Lentmaier, and K. S. Zigangirov, "Some results concerning design and decoding of turbo-codes," *Problems of Information Transmission*, vol. 37, no. 3, pp. 190–205, July-September 2001.

[9] S. Benedetto and G. Montorsi, "Unveiling turbo codes: Some results on parallel concatenated coding schemes," *IEEE Transactions on Information Theory*, vol. 42, no. 2, pp. 409–428, March 1996.

[10] M. Breiling and J. B. Huber, "Upper bound on the minimum distance of turbo codes," *IEEE Transactions on Communications*, vol. 49, no. 5, pp. 808–815, May 2001.

[11] ——, "Combinatorial analysis of the minimum distance of turbo codes," *IEEE Transactions on Information Theory*, vol. 47, no. 7, pp. 2737–2750, Nov 2001.

[12] N. Kahale and R. Urbanke, "On the minimum distance of parallel and serially concatenated codes," submitted to IEEE Transactions on Information Theory. [Online]. Available: http://lthcwww.epfl.ch/~ruediger/papers/weight.ps

[13] L. Bazzi, M. Mahdian, and D. Spielman, "The minimum distance of turbo-like codes." [Online]. Available: http://www-math.mit.edu/~mahdian/tc3.ps

[14] S. Even, A. Pnueli, and A. Lempel, "Permutation graphs and transitive graphs," *Journal of the Association for Computing Machinery*, vol. 19, pp. 400–410, 1972.

[15] G. Solomon and H. van Tilborg, "A connection between block and convolutional codes," *SIAM Journal on Applied Mathematics*, vol. 37, pp. 358–369, October 1979.

[16] C. Berrou, M. Jézéquel, C. Douillard, and S. Kerouédan, "The advantages of non-binary turbo codes," Cairns, Australia, 2001, pp. 61–63.

[17] N. L. Biggs, "Constructions for cubic graphs with large girth," *Electronic Journal of Combinatorics*, vol. 5, 1998.