Al-khwarizmi
Engineering
Journal

Al-Khwarizmi Engineering Journal, Vol.2, No.2, pp 1-14 (2006)

Performance of Turbo-Codes with Some Proposed Interleaver Schemes

Ahmed S. Hadi
Information &Communications Engineering Dept.
AL khwarizmi College of Engineering
University of Baghdad

(Received 14 November 2005; accepted 4 April 2006)

Abstract:-

This paper describes a number of new interleaving strategies based on the golden section. The new interleavers are called golden relative prime interleavers, golden interleavers, and dithered golden interleavers. The latter two approaches involve sorting a real-valued vector derived from the golden section. Random and so-called "spread" interleavers are also considered. Turbo-code performance results are presented and compared for the various interleaving strategies. Of the interleavers considered, the dithered golden interleaver typically provides the best performance, especially for low code rates and large block sizes. The golden relative prime interleaver is shown to work surprisingly well for high puncture rates. These interleavers have excellent spreading properties in general and are thus useful for many applications other than Turbo-codes.

Keywords: FEC, Turbo Code, interleaver, golden section, relative prime.

1. Introduction

Interleaving is a key component of many digital communication systems involving forward error correction (FEC) coding. This is especially true for channels characterized by fading, multipath, and impulse noise, for example. Interleaving, or permuting, of the transmitted elements, provides time diversity for the FEC scheme being employed. An element is used here to refer to any symbol, sample, digit, or bit that is interleaved. In the past the interleaving strategy was usually only weakly linked to the FEC scheme being employed. Exceptions are concatenated FEC schemes such as concatenated Viterbi and Reed-Solomon decoding. The interleaver is placed between the two FEC encoders to help spread out error-bursts and the depth of interleaving is directly linked to the error correction capability of the inner (Viterbi) decoder. More recently, however, interleavers have become an integral part of the coding and decoding strategy itself. Such is the case for Turbo and Turbo-like codes, where the interleaver is a critical part of the coding scheme. The problem of finding optimal interleavers for such schemes is really a code design problem, and is an on-going area of research.

One problem with classical interleavers is that they are usually designed to provide a specific interleaving depth. This is fine if each burst of errors never exceeds the interleaver depth, but it is wasteful if the interleaver is overdesigned (too long) and error-bursts are typically much shorter than the interleaver depth. For example, a simple 10x10 matrix interleaver has an interleaving depth of 10 elements. If a burst of 10 errors occurs, the deinterleaver will optimally spread these 10 errors throughout the block of 100 elements. If the error-burst is 11 elements long, however, then two errors will again be adjacent. If the error-burst is only two elements long then these two errors will only be spaced 10 elements apart after deinterleaving, but they could have been spaced much further apart if it was known that only two errors were present. For example, a 2x50 matrix interleaver would have spaced these two errors 50 elements apart. Of course this interleaver is not good for longer bursts of errors. In practice, most channels usually generate error events of random length, and the average length can be time varying, as well as unknown. This makes it very difficult to design optimum interleaving strategies using the classical approaches. What is sought is an interleaving strategy that is good for any error-burst length.

Section 2 provides some background on Turbo-codes and interleaving methods. Section 3 describes the new interleaving strategies based on the “golden section”. Section 4 compares the bit and packet error-rate performance of Turbo-codes with the various interleavers. Section 5 gives the conclusions.

2. Background

Turbo-codes [1,2,3] have received considerable attention since their introduction in 1993. This is due to their powerful error correcting capability, reasonable complexity, and flexibility in terms of providing different block sizes and code rates. The canonical Turbo-code encoder consists of two 16-state, rate 1/2 recursive systematic convolutional (RSC) encoders operating in parallel with the information bits interleaved between the two encoders, as shown in Figure 1. Without puncturing, the overall code rate is 1/3. This is because the systematic information bits are only sent once. Other code rates can be achieved as required by puncturing the parity bits c_k^1 and c_k^2. It is the job of the interleaver to break apart low-distance error patterns that belong to one RSC code, in the hope that they will create high-distance error patterns in the other RSC code.

A number of close-to-optimum and sub-optimum Turbo decoding methods are possible. The simulation results presented here are based on the enhanced maximum-log-a-posteriori-probability (max-log-APP) approach, with corrected extrinsic information, as described elsewhere in [3, 4,5,6]. It has been found that performance is typically within 0.1 to 0.2 dB of exact, infinite precision log-APP decoding. The amount of degradation is a function of block size, code rate, and signal-to-noise ratio (SNR), with the larger degradations occurring for long blocks, low code rates, and low SNRs. It is convention that one Turbo decoding iteration be defined as two max-log-APP decoding operations.

Interleaving is a key component of any Turbo-code, as shown in Figure 1. Although some form of random or pseudo-random interleaving is usually recommended, it has been found that simple structured interleavers can also offer good performance, especially for short data blocks on the order of a few hundred bits. Examples of common structured block interleavers include relative prime interleavers and $L\times M$ matrix (or block) interleavers using L rows and M columns. An $L\times M$ matrix interleaver is usually implemented by writing into the rows and reading out of the columns, or vice versa. The rows or columns are sometimes read in or out in a permuted order. This permuted order is often implemented using a relative prime. That is, the row or column index can be generated using modulo arithmetic where the index increment and row or column lengths are relative primes. With L or M equal to 1, this type of interleaver simply becomes a one-dimensional relative prime interleaver.

The relative prime interleaver is examined more closely in Section 3.

The original "Turbo" interleaver [1,2] is based on the use of an $M \times M$ matrix with a form of (pseudo-random) relative prime indexing, but the design is much more complicated than that described above. This interleaver has been reported to work well, but is not suited to arbitrary block sizes. This interleaver is not considered further in this paper, but the approach definitely merits further investigation.

Two other interleavers that have been investigated are the "random" interleaver, and the so-called "spread" interleaver [7,8,9]. The random interleaver simply performs a random or pseudo-random permutation of the elements without any restrictions. This interleaver is very useful as a benchmark, and has also been used extensively in calculating error-rate bounds [8,9].

The spread interleaver is really a semi-random interleaver. It is based on the random generation of N integers from 0 to N-1, but with the following constraint [7,9]:

Each randomly selected integer is compared to the S most recently selected integers. If the current selection is within S of at least one of the previous S integers, then it is rejected and a new integer is selected until the previous condition is satisfied.

This process is repeated until all N integers are extracted. The search time increases with S, and there is no guarantee that the process will finish successfully. As a rule of thumb the choice $S < \sqrt{\frac{N}{2}}$ produces a solution in a reasonable amount of time. A number of variations on the spread interleaver are presented in [10,11,12,13]. These variations are not considered further here, but also merit further investigation.

3. Golden Section Interleaving

3.1 The Golden Section

The golden section arises in many interesting mathematical problems. Figure 2 illustrates the golden section principle in relation to the interleaving problem of interest. Given a line segment of length 1, the problem is to divide it into a long segment of length g, and a shorter segment of length 1-g, such that the ratio of the longer segment to the entire segment is the same as the ratio of the shorter segment to the longer segment.
That is,

$$\frac{g}{1} = \frac{1-g}{g} \qquad (1)$$

$$g = \frac{\sqrt{5}-1}{2} = 0.618 \qquad (2)$$

Now consider points generated by starting at 0 and adding increments of g, using modulo-1 arithmetic. After the first increment there are two points at 0 and g that are 1-g apart, using modulo-1 arithmetic. Modulo distances are used to allow for the option of having the first point start anywhere along the line segment. From (1), the distance of 1-g is the same as g^2 . After the second increment the first and third points determine the minimum distance and this distance is g^3. Again, this follows from the definition of g in (1). After the third increment the first and fourth points determine the minimum distance and this distance is g^4. The minimum distance after the fifth point is the same. The minimum distance after the sixth point is g^5. This trend continues, with the minimum distance never decreasing by more than a factor of g when it does decrease. This property follows directly from the definition of the golden section in (1). The same distances can also be generated with the complement increment of $(1-g)=g^2 \approx 0.382$. Higher powers of g can also be used for the increment value, but the initial minimum distances are reduced to the smaller increment value.

Figure 3 shows a plot of the minimum distances versus the number of points considered, as points are added using an increment of g with modulo-1 arithmetic.

Figure 3 also shows an upper bound for each specific number of points. That is, given n points, and only n points, they could be uniformly spaced with a minimum distance of 1/n. Of course the golden section increment results are valid for all numbers of points at the same time. The upper bound is not. Even so, the golden section increment results are seen to track the upper bound quite closely. Note that even when the minimum distance drops, most points will still be at the previous minimum distance from their neighbours, with the average distance between points equal to the upper bound.

The distance properties of the golden section increment, illustrated in Figure 3, are desirable for interleavers in general, but in particular are desirable for Turbo-code interleavers. It is now shown how these properties can be used in designing a number of practical interleavers.

3.2 Golden Relative Prime Interleavers

For golden relative prime interleavers, the interleaver indexes are calculated as follows:

$$\alpha(i) = (s + i.t) \bmod L\_info, i=0 \ldots L\_info-1 \qquad (3)$$

where s is an integer starting index, t is an integer index increment, and L_info is the interleaver length. L_info and t must be relative primes to ensure that each element is read out once and only once. The starting index s is usually set to 0, but increment, s, is chosen “close” (as further defined below) to one of the non-integer values of

$$c = L\_info(g^{m} + j)/r \qquad (4)$$

where g is the golden section value, m is any positive integer greater than zero, r is the index spacing (distance) between nearby elements to be maximally spread, and j is any integer modulo r. The preferred values for m are typically 1 or 2. In a typical implementation where adjacent elements are to be maximally spread, j is set to 0 and r is set to 1. For Turbo-codes, however, greater values of j and r can be used to obtain the best spreading for elements spaced r apart. For example, r could be set to the repetition period of the feedback polynomial in the RSC encoder, to maximally spread input-weight-2 error events.

One definition of being a “close” relative prime is to fall within a small window about the exact real value, c, given in (4). The simplest choice is to select the relative prime t closest to c, for predetermined values of L_info, m, j, and r. The result is a golden relative prime interleaver with quantization error. For large blocks the quantization error is usually not significant for short error-burst lengths, but can grow to be significant after many increments. One way to mitigate the quantization error problem is to perform a search for the best relative prime increment t in the vicinity of c, by using the minimum distance between interleaved indexes for the maximum number of elements considered, as a measure of the spreading quality of the interleaver. Alternatively, the best relative prime increment, t, in the vicinity of c, is determined by the sum (or weighted sum) of the minimum distances between interleaved indexes for all numbers from two up to the maximum number of elements considered. In this case, the best choice for t is that which maximizes the area under the minimum distance curve.

Figure 4 shows the spreading properties for an interleaver having a size L_info=1028 (e.g. used in a Turbo-code encoder with 1024 information bits and 4 flush bits per block), m=2, j=0, r=1, and a relative prime increment of t=393. The value of $c = L\_info \times g^{2}$ is approximately 392.7. The value of p=393 is the closest relative prime. As can be seen, this golden relative prime interleaver performs well in tracking the upper bound near the origin, but does not appear to be as good away from the origin where the accumulating quantization error becomes significant. The area under the entire curve is 4620. This spreading measure is used for comparison purposes below.

Most general purpose digital signal processors (DSPs) today offer the kind of modulo indexing indicated in (3) to implement circular buffers. It is also trivial to implement in hardware. Thus, golden relative prime interleavers require no additional memory and little or no additional processing compared to that required to store and read an uninterleaved vector.

3.3 Golden Interleavers

Golden interleavers do not use integer relative primes and integer modulo arithmetic, but rather are based on sorting real-valued numbers derived from the golden section. The first step is to compute the golden section value g. The second step is to compute the real increment value c, as defined previously in (4). The third step is to generate real-valued golden vector v. The elements of **v** are calculated as follows:

v(i)=(s + i. c),mod L_info,i=0…L_info-1 (5)

where s is any real starting value. The next step is to sort golden vector **v** and find the index vector **z** that defines this sort. That is, find sort vector **z** such that a(i)=v(z(i)), i=0…L_info-1, where a=sort(v). The golden interleaver indexes are then given by α(z(i))=i, i=0…L_info-1. In fact, vector **z** is the inverse interleaver for α.

The starting value s is usually set to 0, but other real values of s can be selected. The preferred values for m are typically 1 or 2, as discussed previously. For maximum spreading of adjacent elements, j is set to 0 and r is set to 1. For Turbo-codes, greater values of j and r may be used to obtain the best spreading for elements spaced r apart, as discussed previously.

The golden interleaver does not suffer from accumulating quantization errors, as does the golden relative prime interleaver. In the golden interleaver case, a quantization error only occurs in the final assignment of the indexes. On the other hand, the golden interleaver cannot be implemented using the simple modulo-increment indexing method described above for the golden relative prime interleaver. In contrast, the golden interleaver indexes must be pre-computed and stored in index memory for each block size of interest. If the full indexes are stored, then the index memory can be excessive. For example, an interleaver of length 2^{16} elements would require 16×2^{16} bits of index memory. The required amount of index memory can be significantly reduced by only storing index offsets. For example, the n-th index can be ea sily calculated as required using α(i)=floor[v(i)]+o(i), where the floor function extracts the integer part, v(n) is calculated using real mod L_info arithmetic as in (5), and by definition o(n) is the required index offset stored in index memory. The number of bits that are required to store each index offset is typically only one or two. Thus, for the example above, the index memory is reduced to 2×2^{16} bits, or about 1/8 that required for full storage of the indexes.

Figure 5 shows the spreading properties for a golden interleaver having size L_info=1028, m=2, j=0, and r=1. The value of real increment c= L_info$\times$g^{2} is approximately 392.7. As can be seen from Figure 5, the golden interleaver performs very well in tracking the theoretical upper bound, and tracks it better than the golden relative prime interleaver curve shown in Figure 4. Note that the area under the curve has increased from 4620, for the golden relative prime interleaver, to 5250, for the golden interleaver, indicating that the golden interleaver is better at spreading out error-bursts of arbitrary length.

3.4 Dithered Golden Interleavers

It has been found for Turbo-codes that interleavers with some randomness tend to perform better than completely structured interleavers, especially for large block sizes on the order of 1000 or more bits. However, the spreading properties of the golden interleaver are still very desirable, both to maintain a good minimum distance (a steep error curve at high SNRs) and to ensure rapid convergence by efficiently spreading error-bursts throughout the block. These two features are encompassed in the dithered golden interleaver. The only difference between the golden interleaver and the dithered golden interleaver is the inclusion of a real perturbation (dither) vector d, in golden vector v. That is,

v(i)=(s+i.c+d(i),mod L_info,i=0…L_info-1, (6)

where d(i) is the i-th dither component. The added dither is uniformly distributed between 0 and L_info $\times$D, where D is the normalized width of the dither distribution. The dithered golden vector **v** is sorted, and interleaver indexes are generated in a similar manner to that for the golden interleaver described above.

It has been found experimentally, for Turbo-codes, that a crude rule of thumb for any block size is to use D$\approx$0.01. The result is that for small blocks, on the order of 1000 bits or less, the effect of the dither component is small. For large blocks, on the order of 1000 bits or more, the effect of the dither component naturally increases as the block size increases. In practice, the optimum amount of dither for a specific Turbo-code is a function of the block size and the code rate obtained with puncturing.

Similar to the golden interleaver, the dithered golden interleaver requires the use of index memory for storing pre-computed indexes, and therefore cannot be implemented using the simpler method of modulo-increment indexing. As for the golden interleaver, the required amount of index memory can be significantly reduced by only storing index offsets. The amount of memory required now depends on the degree of dither, and whether the dither component is included in the calculation of each approximate index, or whether it is totally accounted for in the stored index offset.

In conclusion, the dithered golden interleaver maintains most of the desirable spreading properties of the golden interleaver, but is also capable of adding randomness to the interleaver to improve Turbo-code performance. Further, the golden interleaver is now just a special case of the dithered golden interleaver with D=0.

3.5 Performance Results

Performance results are presented for a fixed interleaver size of L_info=1028 (historically selected for a 1024 info-bit block with 4 flush bits). The Turbo-code uses two identical, parallel, 16-state, rate 1/2 RSC codes, with polynomials $(23,35)_8$. The repetition period of the feedback polynomial is r=15. Results are presented for nominal code rates of 1/3 (unpunctured), 1/2 and 4/5. The Turbo decoding method used is the enhanced max-log-APP (a posterior probability) approach presented in [5,6]. This method typically provides performance with 0.1 to 0.2 dB of exact, infinite precision APP processing. The maximum number of decoding iterations was set to 16. A simple early stopping criterion was used, which helped speed up the simulations [5]. A more extensive list of the encoder and decoder specifications is given in [6].

The RSC code trellis termination method is critical to the performance of Turbo-codes, especially with good interleavers. A number of generally applicable dual-termination and dual-tail-biting techniques are presented in [14,15]. These termination techniques do not place any restrictions on the interleaver design. The recommended approach for large blocks (>1000 info-bits) is to perform dual-termination. Without termination, both RSC encoders start in the zero-state and both stop in an unknown state. With single-termination, a commonly used approach in the literature, the interleaver includes 4 flush bits, both RSC encoders start in the zero-state, and one RSC encoder is known to stop in the zero-state. With dual-termination, the interleaver includes 8 flush bits and both RSC encoders are known to start and stop in the zero-state. For large blocks the flush overhead is negligible.

Figure 6 shows the packet error rate (PER) performance for rate 1/2 codes, with the three termination options mentioned above. The same dithered golden interleaver, with m=1, j=0, r=1, and D=0.02, was used in all three cases. This figure clearly shows the importance of proper trellis termination, when a good interleaver is used. The conclusion is quite different for a random interleaver. The bit error rate (BER) results corresponding to Figure 6 are shown in Figure 7. The remaining results were all obtained with dual-termination.

Figure 8 shows the PER results for a code rate of 1/3, and four different interleavers of size L_info=1028. The interleavers used are the “random” interleaver, the relative “prime” interleaver with t=393 (closest relative prime to $L\_info \times g^2$), the “spread” interleaver with spread parameter S=18, and the dithered “golden” interleaver with m=1, j=0, r=1, and D=0.02. The spread interleaver address generator was obtained from [12]. As expected, the highly structured relative prime interleaver does not perform well at high SNRs due to its inability to break up coupled error events. It does, however, do an excellent job of eliminating the low-distance input-weight-2 events (multiples of 15 to multiples of 15). Note that the random interleaver is not much better for this block length. The spread interleaver offers a significant improvement, but the dithered golden interleaver provides the best performance. Figure 9 shows the corresponding BER results. It is worth noting that the BER results for the random interleaver, at high SNRs, agree quite closely with the theoretical bounds presented in [8,9].

Concerning statistical reliability, 1000 packet errors were counted in the upper portion of each curve. The goal for the lowest point on each curve was to count on the order of 100 packet errors. The least reliable result is for the lowest point on Figures 8 and 9, for which only 20 packet errors were counted. Even so, it is safe to say that the “bend” in the BER curve, for the dithered golden interleaver, is in the vicinity of 10^{-10} .

Figure 10 shows the PER results for a punctured code rate of 4/5, and the same four interleaver types. The random, relative prime, and spread interleavers were exactly the same as before. The best parameters found for the dithered golden interleaver were m=1, j=9, r=15, and D=0.005. The dithered golden interleaver is again the best. What is somewhat surprising is how well the relative prime interleaver performs. This is partly explained by the fact that, for high puncture rates, it becomes more important to eliminate low-distance input-weight-2 events, and the relative prime interleaver is ideally suited to this task. The spread interleaver is not as effective at eliminating such events, and therefore performance is degraded. Figure 11 shows the corresponding BER results. Note that the bend in the BER curve occurs much higher for highly punctured codes. Even so, this high rate code, with a dithered golden interleaver, still provides excellent performance for BERs down to 10^{-7} .

4.Conclusions

1. Three new interleavers based on the golden section were presented. They are called the golden relative prime interleaver, the golden interleaver, and the dithered golden interleaver. Random and spread interleavers were also considered. Turbo-code performance results were presented and compared for the various interleavers.

2. The dithered golden interleaver provided the best performance in all cases considered.

3. The golden relative prime interleaver, although highly structured with no random component, worked surprisingly well for high code rates.

4. Using a dithered golden interleaver of size L_info=1028, it was shown that a parallel, dual-terminated, 16-state, rate 1/3 Turbo-code can achieve a BER of 10^{-10} at an Eb/N0 value of 1.6 dB. The bend in the BER curve also occurs at a BER of about 10^{-10} . Puncturing this same code to rate 4/5 moved the bend out and up to a BER of about 10^{-7} . Further improvements should be possible by incorporating more specific knowledge about the punctured component RSC codes into the interleaver design.

5. The various “golden” interleavers have excellent spreading properties in general and

are thus useful for many applications other than Turbo-codes

6. There are no restrictions on the block size, and a time-consuming search is not required. Thus, interleavers can be easily generated on an as needed basis for any block length.

5.References

[1] Ber 1993, C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon Limit Error-Correcting Coding and Decoding: Turbo-Codes", Proceedings of ICC'93, Geneva, Switzerland, pp. 1064-1070, May, 1993.

[2] Ber 1996, C. Berrou, and A. Glavieux, "Near Optimum Error Correcting Coding and Decoding: Turbo-Codes", IEEE Trans. On Comm., Vol. 44, No. 10, October 1996. [3] B. Talibart and C. Berrou, "Notice Preliminaire du Circuit Turbo-Codeur/Decodeur TURBO4", Version 0.0, June, 1995.

[3] Aba 2002, N. H. Abbas, "Turbo Code Design for Wireless Communications," Master's Thesis, University of Baghdad, Iraq, November 2002.

[4] Cro 1998, S. Crozier, A. Hunt, K. Gracie, and J. Lodge, "Performance and Complexity Comparison of Block Turbo-Codes, Hyper-Codes, and Tail-Biting Convolutional Codes", 19-th Biennial Symposium on Communications, Kingston, Ontario, Canada, pp.84-88, May 31-June 3, 1998.

[5] Gra 1998, K. Gracie, S. Crozier, A. Hunt, and J. Lodge, "Performance of a Low-Complexity Turbo Decoder and its Implementation on a Low-Cost, 16-Bit Fixed-Point DSP", The 10-th International Conference on Wireless Communications (Wireless'98), Calgary, Alberta, Canada, pp.229-238, July 6-8, 1998.

[6] Gra 1999, K. Gracie, S. Crozier, and A. Hunt, "Performance of a Low-Complexity Turbo Decoder with a Simple Early Stopping Criterion Implemented on a SHARC Processor", International Mobile Satellite Conference (IMSC'99), Ottawa, Canada, June 16-18, 1999.

[7] Hun 1999, A. Hunt, S. Crozier, M. Richards, and K. Gracie, "Performance Degradation as a Function of Overlap Depth when using Sub-Block Processing in the Decoding of Turbo Codes", International Mobile Satellite Conference (IMSC'99), Ottawa, Canada, June 16-18, 1999.

[8] Div 1995, D. Divsalar and F. Pollara, "Multiple Turbo Codes for Deep-Space Communications", JPL, TDA Progress Report 42-121, May 15, 1995.

[9] Ben 1996, S. Benedetto and G. Montorsi, "Unveiling Turbo Codes: Some Results on Parallel Concatenated Coding Schemes", IEEE Trans. on Inform. Theory, Vol. 42, No. 2, pp.409-428, March 1996.

[10] Ben 1997, S. Benedetto and G. Montorsi, "Tutorial 11: Turbo Codes: Comprehension, Performance Analysis, Design, Iterative Decoding", IEEE International Conference on Communications (ICC'97), Montreal, Quebec, Canada, June 8-12, 1997.

[11] Bar 1994, A. Barbulescu and S. Pietrobon, "Interleaver Design for Turbo Codes", Electronics Letters, Vol. 30, No. 25, pp.2107-08, December 8, 1994.

[12] Ho 1998, M. Ho, S. Pietrobon, and T. Giles, "Interleavers for Punctured Turbo Codes", IEEE Asia-Pacific Conf. on Commun. (APCC'98) and Singapore Int. Conf. on Commun. Systems, Vol. 2, pp. 520-524, Singapore, November, 1998.

[13] Pit 1998, S. Pietrobon, "Interleaver Address Generator", Version 1.01, October 4, 1998, available from Small World Communications. See www.sworld.com.au.

[14] Sau 1998, P.-P. Sauvé, "Multibit Decoding of Turbo Codes," Master's Thesis, University of Toronto, Canada, October 1998.

[15] Gui 1994, P. Guinand and J. Lodge, "Trellis Termination for Turbo Encoders", Proc. 17th Biennial Symp. On Communications, Queen's University, Kingston, Canada, pp. 389-392, May 30-June 1, 1994.

.

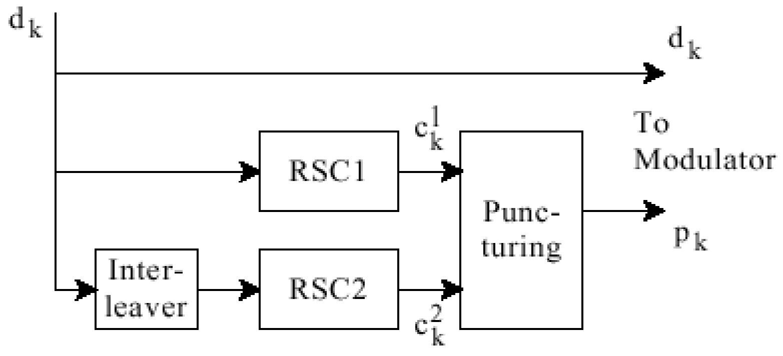

Fig. 1. Turbo-code encoder using two rate 1/2 RSC codes with puncturing.

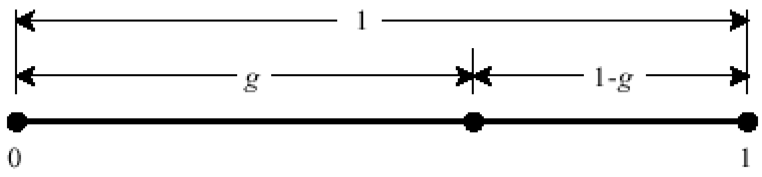

Fig. 2. Illustration of the golden section principle.

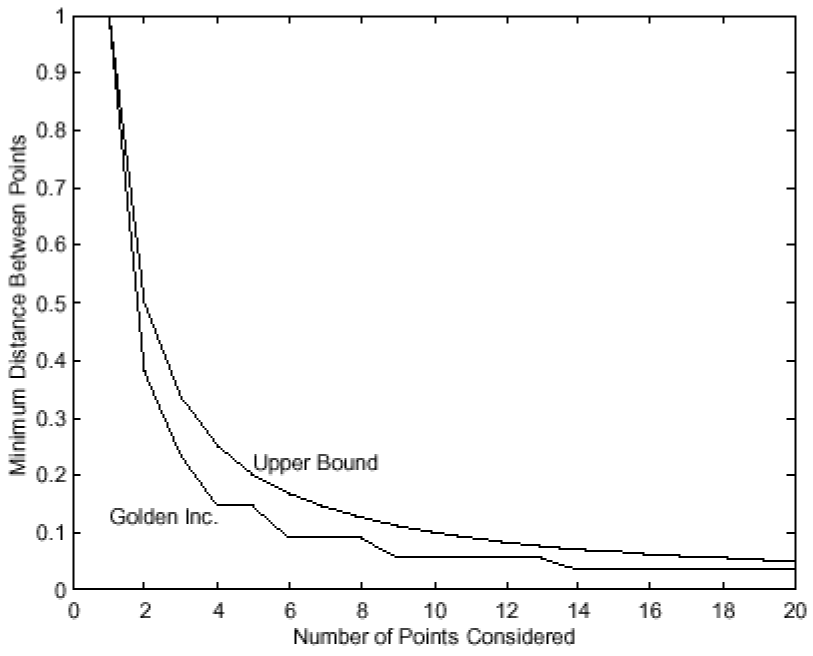

Fig. 3. Minimum distance between points versus number of points with a golden increment.

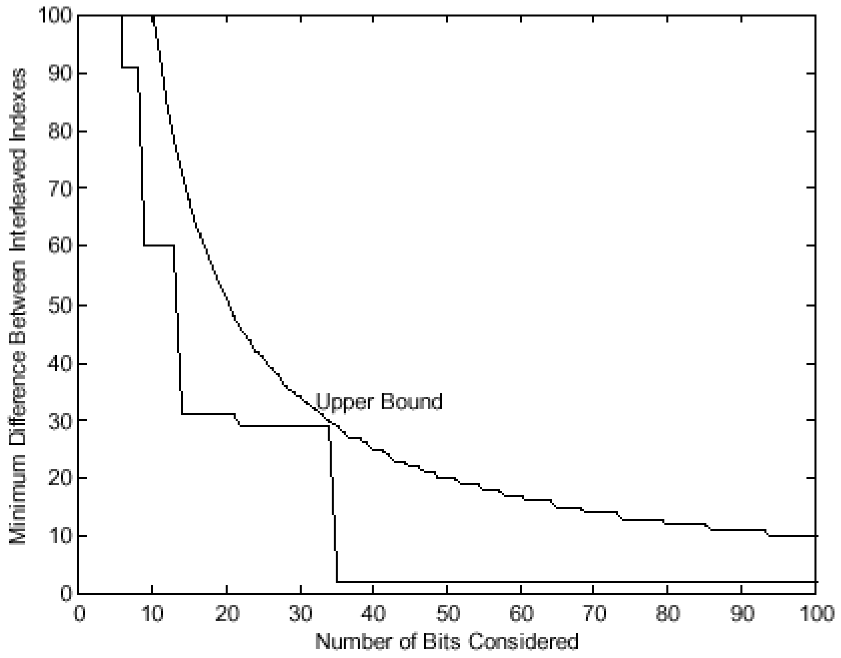

Fig. 4. Minimum difference between interleaved indexes versus number of bits considered with a golden relative prime interleaver. L_info=1028, t=393, area under curve=4620.

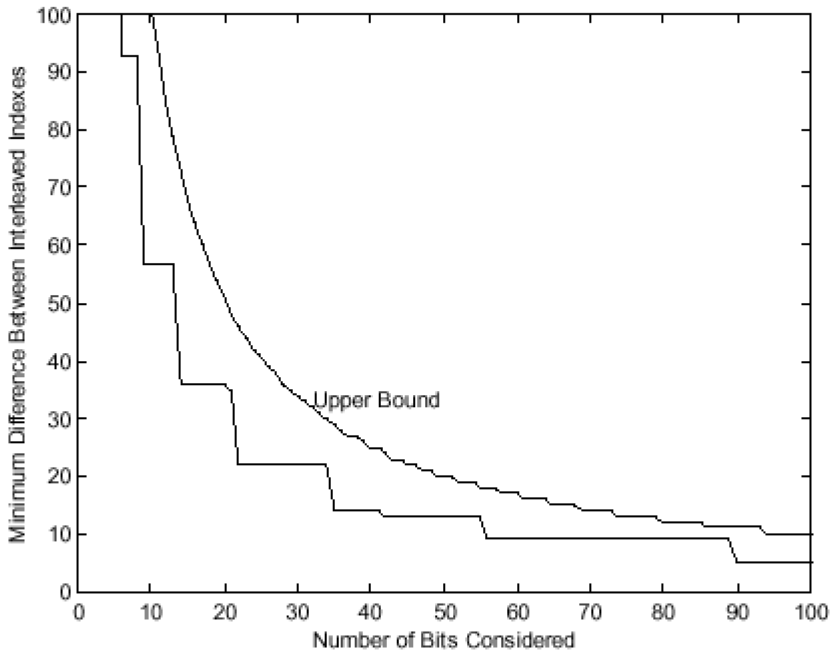

Fig. 5. Minimum difference between interleaved indexes versus number of bits considered with a golden interleaver. L_info=1028, m=2, j=0, r=1, area under curve=5250.

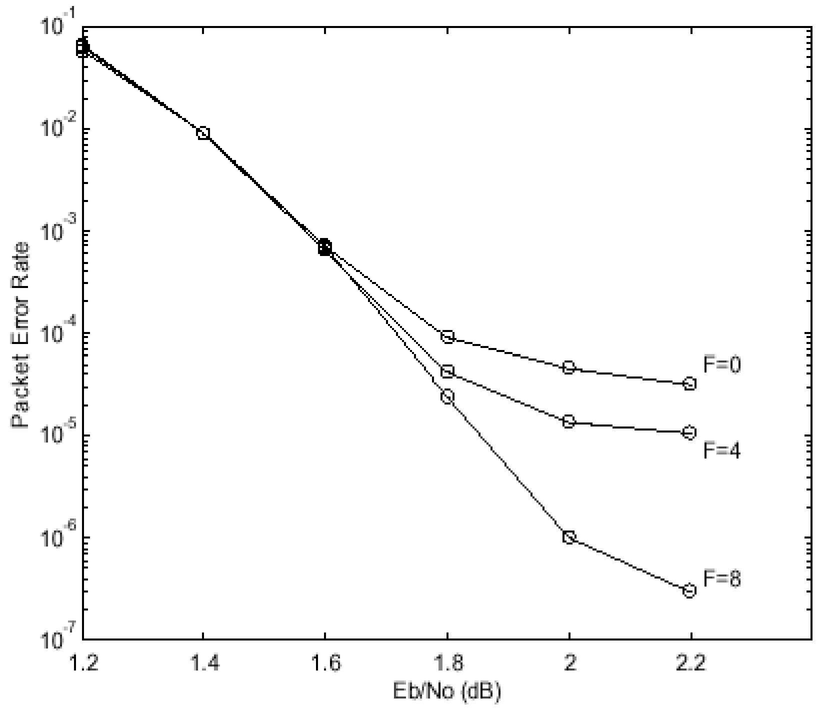

Fig. 6. PER performance for rate 1/2 codes and a dithered golden interleaver with L_info=1028, m=1, j=0, r=1, and D=0.02. Results are shown for no-termination (number of flush bits F=0), single-termination (F=4), and dual-termination (F=8).

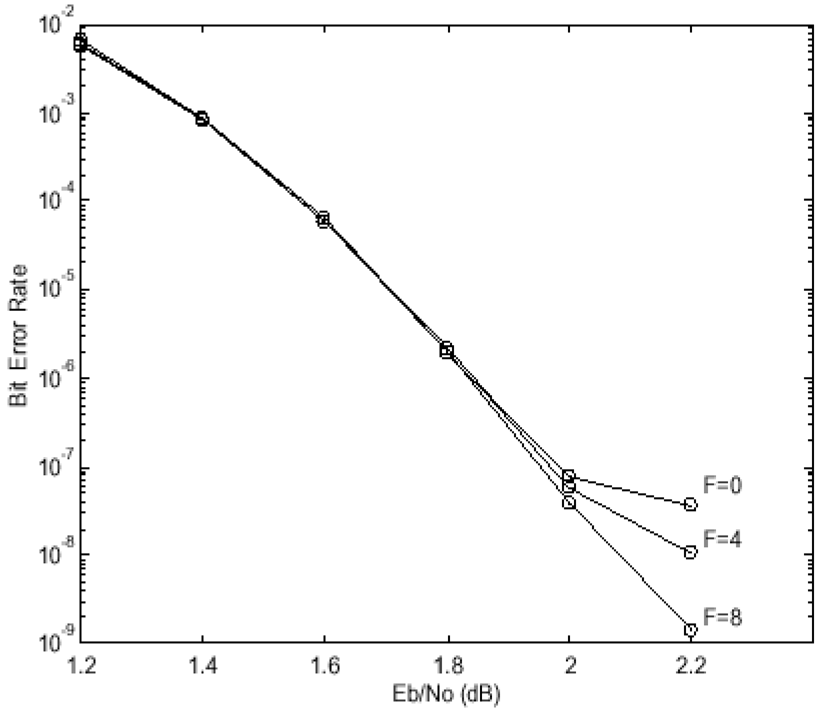

Fig. 7. BER performance for rate 1/2 codes and a dithered golden interleaver with L_info=1028,m=1, j=0, r=1, and D=0.02. Results are shown for no-termination (number of flush bits F=0), single-termination (F=4), and dual-termination (F=8).

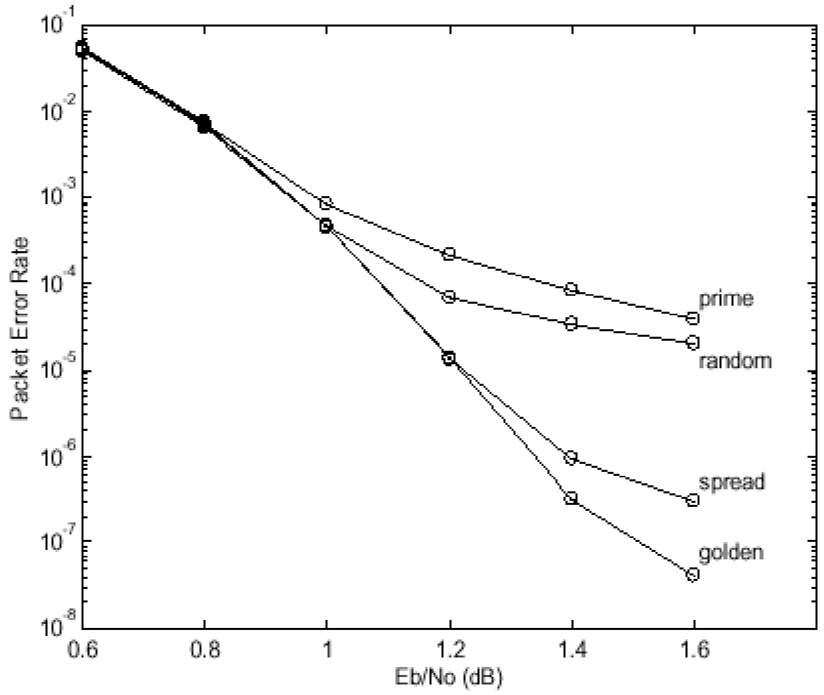

Fig. 8.PER performance for rate 1/3 codes with dual-termination and L_info=1028. The interleavers used are the “random” interleaver, the relative “prime” interleaver (p=393), the “spread” interleaver (S=18), and the dithered “golden” interleaver (m=1, j=0, r=1,

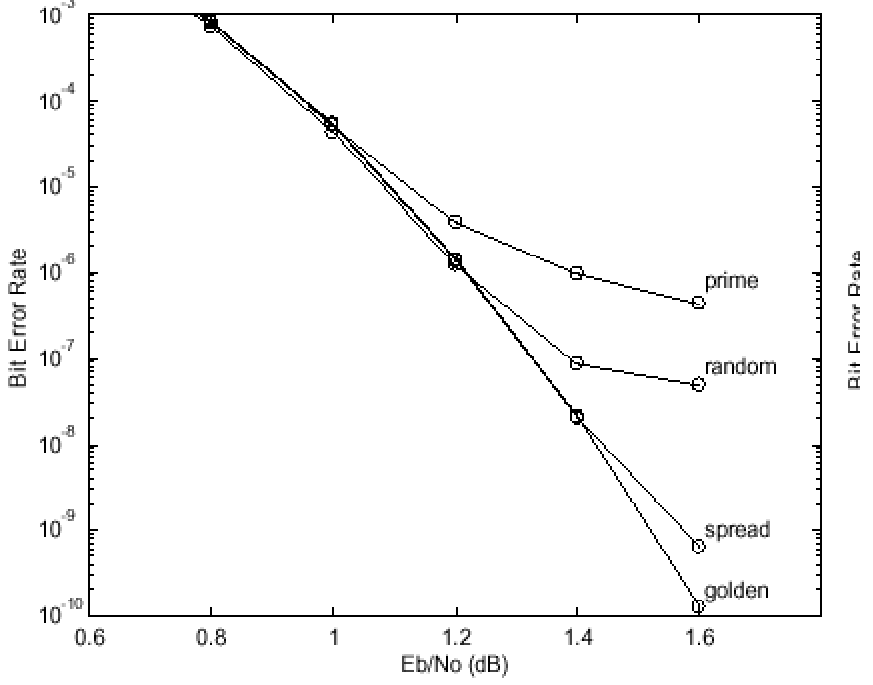

Fig. 9. BER performance for rate 1/3 codes with dual-termination and L_info=1028. The interleavers used are the “random” interleaver, the relative “prime” interleaver (t=393), the “spread” interleaver (S=18), and the dithered “golden” interleaver (m=1, j=0, r=1, and D=0.02).

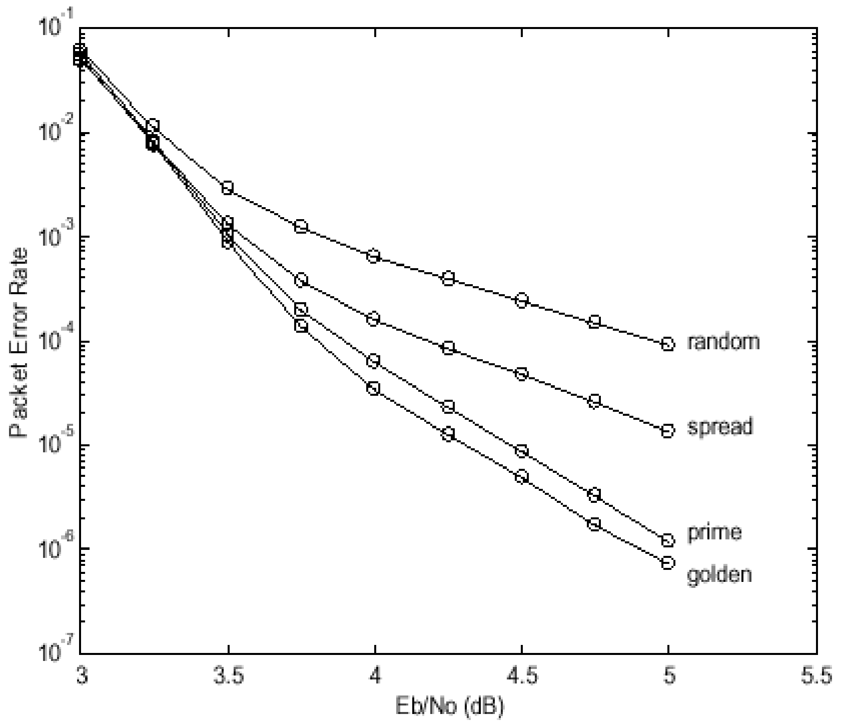

Fig. 10.PER performance for rate 4/5 codes with dual-termination and L_info=1028. The interleavers used are the “random” interleaver, the relative “prime” interleaver (t=393), the “spread” interleaver *(*S=18), and the dithered “golden” interleaver *(*m=1, j=9, r=15, and D=0.005).

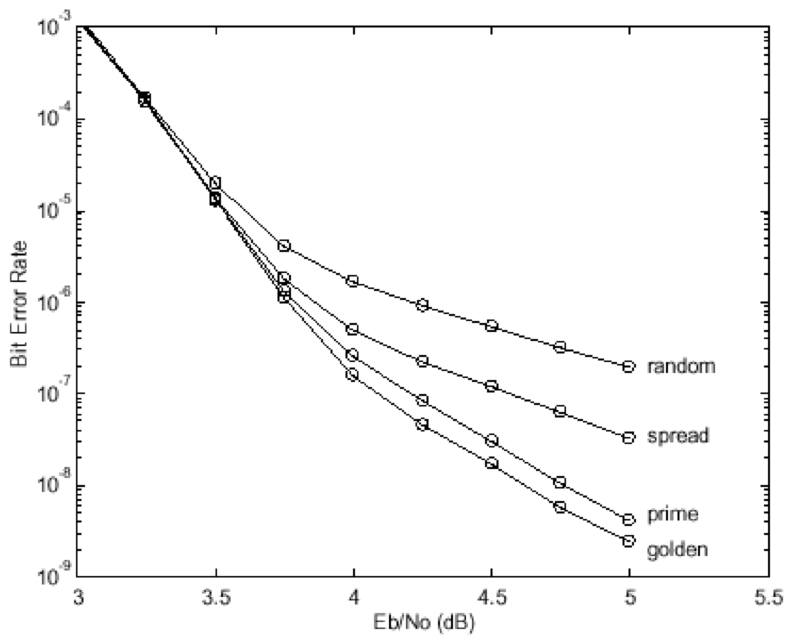

Fig. 11 BER performance for rate 4/5 codes with dual-termination and L_info=1028. The interleavers used are the “random” interleaver, the relative “prime” interleaver (t=393), the “spread” interleaver *(*S=18), and the dithered “golden” interleaver *(*m=1, j=9, r=15, and D=0.005).

أداء الجفرة المسرعة مع بعض المفرقات المقترحة

أحمد ستار هادي
قسم هندسة المعلومات والاتصالات
كلية هندسة الخوارزمي/ جامعة بغداد

الخلاصة :
تصف هذه المنشورة عدد من استراتيجيات التفريق (interleaved) الجديدة المستندة على المقطع الذهبي المفرقات الجديدة تدعى البارز النسبي الذهبي (golden relative interleaver) والذهبي (golden) والمرتبك الذهبي (dithered golden) . الطريقتان الثانية والثالثة تتضمن قيم حقيقية مرتبة مشتقة من المقطع الذهبي . المفرق العشوائي الذي عادة يسمى مفرق الانتشار (spread) أيضا سيؤخذ بنظر الاعتبار . نتائج اداء الجفرة المسرعة سوف تبين وتقارن مع عدد من استراتيجيات التفريق . المفرق المرتبك يكون اداه الافضل نموذجيا وبصورة خاصة في حالة نسبة الجفرة (code rate) القلية وحجم الكتلة الكبير . المفرق البارز النسبي الذهبي يكون اداة مدهش في نسب الثقب (puncture) العالية . هذه المفرقات لها خصائص انتشار مفيدة ومستخدمة في عدة تطبيقات غير الجفرات المسرعة.