Turbo codes optimization using genetic algorithms

Nicolas Durand, Jean-Marc Alliot, Boris Bartolome

Turbo Codes Optimization Using Genetic Algorithms

Nicolas Durand
Laboratoire d'Optimisation Globale
Centre d'Etudes de la Navigation Aérienne
7, av Edouard Belin
31055 Toulouse cedex France
durand@cenatls.cena.dgac.fr

Jean-Marc Alliot
Laboratoire d'Optimisation Globale
Centre d'Etudes de la Navigation Aérienne
7, av Edouard Belin
31055 Toulouse cedex France
alliot@recherche.enac.fr

Boris Bartolomé
Ecole Nationale Supérieure des Télécommunications
10, av Edouard Belin
BP: 4004, 31028 Toulouse cedex 04 France
Boris.Bartolome@supaero.fr

1 Introduction

The evolving world of telecommunications requires an increasing reliability and speed in communications. Reliability in information stocking and transmission is provided by coding techniques. Information is usually coded and stocked in bit strings. The communication medium can generate errors that have to be detected and corrected when decoding. Therefore, more bits than required are transmitted in order to check the errors due to the communication medium. The code rate can be defined as the information bit string length divided by the number of transmitted bits.

A simple way to code can be to assign to any incoming word (or bit string) a longer fixed coded word; decoding can then be very long as a transmitted coded word must be associated to the nearest (according to the Hamming distance) existing coded word. This technique is called the maximum likelihood technique. Consequently, the larger the Hamming distance between any pair of coded words, the easier the decision on a received coded word. For linear codes (generally used), the minimum Hamming distance between any pair of coded words (also called the *Free Distance* of a code) is also the minimum Hamming distance between the zero coded word (00...0) and any non zero coded word.

One of the challenging problems today is to find a coding technique that maximizes the Free Distance of a code. The problem is very difficult as in real applications, the Free Distance of a code cannot be measured exactly. After introducing the coding techniques, a Genetic Algorithm is detailed in order to optimize the interleaving matrix of a Parallelly Concatenated Recursive Convolutive Turbo Code (PCRCTC). Results on a problem of size 105 are compared to existing interleaving matrices.

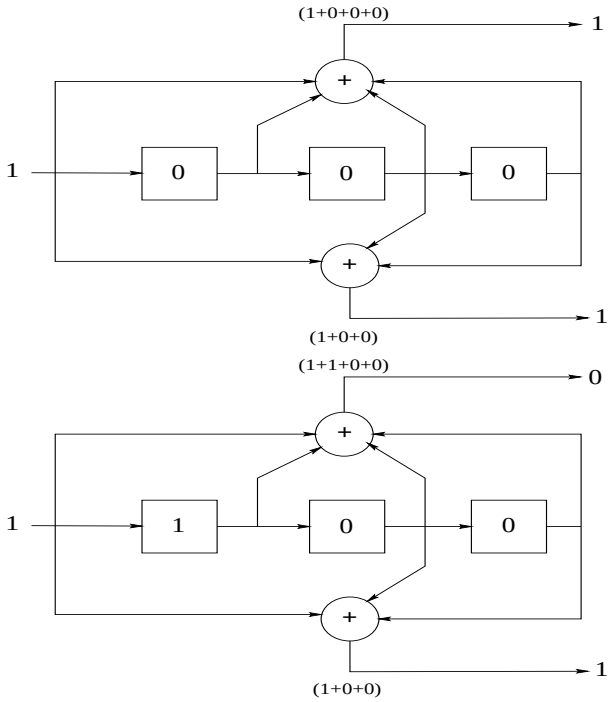

Figure 1 Sample of forward convolutive coder: first and second step - input: $1-1$ **- output:** $(11)-(01)$

2 Coding techniques

Coding techniques could be divided, until recently, into two main classes: convolutive and block codes. The introduction of PCRCTC, (C.Berrou and Thitimajshima 1993) in 1993 erased this border by mixing the two methods and showing astonishing performances. Sections (2.1) to (2.5) give the general schemes of different coding techniques.

2.1 Forward Convolutive Coding

Figure (1) gives the general scheme of a simple forward convolutive coder.

At each clock tick, a bit enters the coder, additions on bits are done according to figure (1). In this example, the rate of the coder is $\frac{1}{2}$. Bits enter the coder one by one and are not grouped.

2.2 Recursive Convolutive Coding

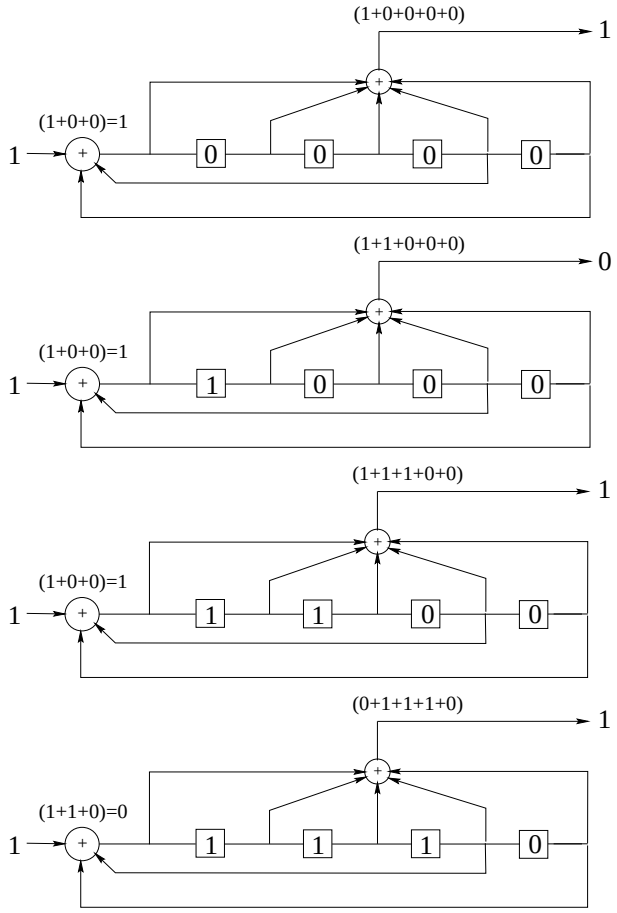

Figure 2 Sample of recursive convolutive coder: input string ($1-1-1-1$) - output string ($1-0-1-1$)

Figure (2) gives the general scheme of a recursive convolutive coder. The coded bits at time t depend on all former information bits, whereas in section (2.1) they didn't. This is a very important property for the turbo decoding algorithm.

2.3 Systematic Recursive Convolutive Coding

Figure (3) gives an example of a systematic recursive convolutive coder. A systematic coder also sends the uncoded bit along with the coded bits. This is also a very important property for the decoding algorithm.

2.4 Block Codes

Block codes are linear codes in which bits are grouped and a linear transformation is applied to all the block. Block codes can usually be described with a linear application (interleaving matrix), but it is not necessary.

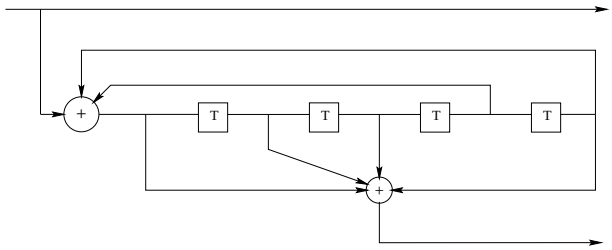

Figure 3 Sample of systematic recursive convolutive coder

2.5 Parallely Concatenated Systematic Recursive Convolutive Coding

An example of parallely concatenated systematic recursive convolutive coder is shown on figure (4).

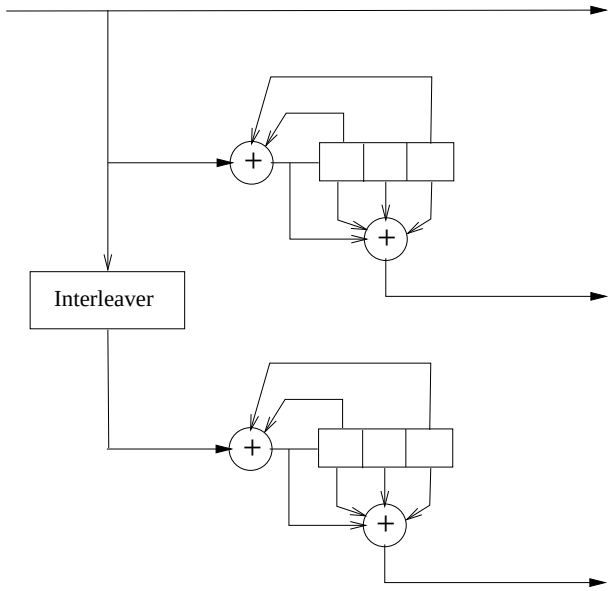

Figure 4 Sample of parallely concatenated systematic recursive convolutive coder

An interleaving matrix has been added in between the two coders. If the size of the interleaving matrix is N, then the bits which enter the coder are grouped in blocks of size N and:

- are directly transmitted (because of systematicity);
- are coded with the first recursive convolutive coder and are then transmitted ;
- are scrambled by the interleaver (which is simply a permuter) and are then coded with the second recursive convolutive coder before being transmitted.

The interleaver matrix is one of the key pieces in this scheme because it determines the Free Distance of the overall code, and therefore the performances of the code. This is the

scheme used to code Parallely Concatenated Recursive Systematic Turbo Codes. Properties from convolutive coders (the individual coders) and block coders (because of the interleaving matrix) are mixed. The coding rate is $\frac{1}{3}$. The aim of this article is to introduce a technique to optimize the interleaving matrix in order to maximize the Free Distance of the code.

The following definitions will help the understanding of the paper:

Worst Vector: a *worst vector* is a vector that leads to the minimum Free Distance or more precisely Parallel Convolutive Turbo Codes distance of a given matrix.

Self Terminating vector: a vector is *self-terminating* if it allows the *first* encoder (that is, with no interleaving) to end in state zero.

2.6 Transmission Channel

Figure (5) gives a very simplified scheme of a general transmission. In the following it was supposed that the

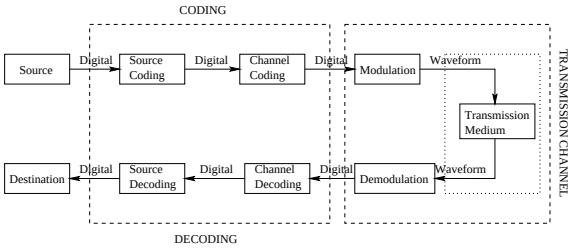

Figure 5 Simplified scheme of a standard transmission

modulation-demodulation operation did not affect the transmitted signal (a perfect synchronization and demodulation is supposed) and the Additive White Gaussian Noise (AWGN) channel was chosen. This channel adds noise over every transmitted bit, and the noise is a centered gaussian random variable whose variance depends upon the energy of the transmitted bit.

2.7 The Turbo Decoding Principle

In this article, the Turbo Decoding principle will not be detailed because of its complexity. People interested in decoding will find a very detailed presentation in (Robertson 1994). A decoding engine was built to show the benefit of improving the Free Distance on the example detailed in part (4).

2.8 Why maximizing the Free Distance

The asymptotic performance for a Turbo Code approaches (Lance C. Perez and Jr. 1996):

$$P_b \approx \frac{N_{free}\tilde{\omega}_{free}}{N} Q\left(\sqrt{d_{free}\frac{2R_C E_b}{N_0}}\right) \quad (1)$$

where:

- N_{free} is the multiplicity of the codewords giving the Free Distance of the code
- $\tilde{\omega}_{free}$ is the mean Hamming weight of the words giving the Free Distance of the code
- N is the interleaver size
- Q equals $Q(x) = \frac{1}{\sqrt{2\pi}} \int_x^{+\infty} e^{-\frac{y^2}{2}} dy$
- d_{free} is the Free Distance of the code
- R_C is the rate of the code ($\frac{1}{3}$ in our case)
- E_b is the bit energy
- N_0 is the spectral power density for the white gaussian noise

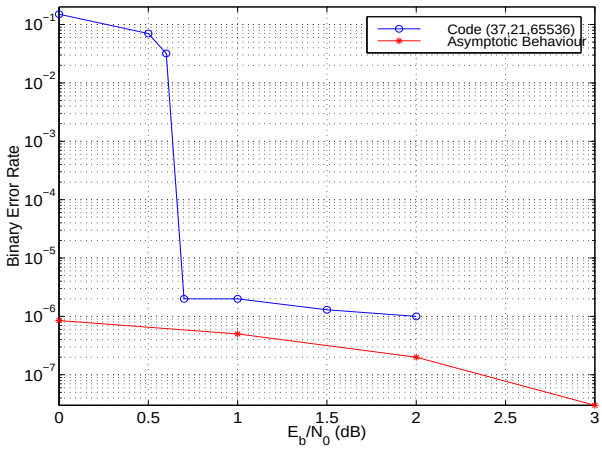

Figure 6 Example of floor phenomenon

An example of asymptotic curve is shown on figure (6). Increasing the Free Distance of the code will improve its asymptotic performance beyond about a Bit Error Rate (BER) of 10^{-6}.

3 Maximizing the Free Distance using GAs

In this paper, classical Genetic Algorithms and Evolutionary Computation principles such as described in the literature (Goldberg 1989, Michalewicz 1992) are used.

The algorithm is first described for an interleaving depth of 105 bits; adaptations to higher interleaving depths are then discussed.

A depth length of 105 bits was chosen because:

1. In (Maseng 1997), an impressive algorithm is detailed to improve the Free Distance of a PCSRCC with a depth length of 105 bits.
2. This same paper compares, in a very detailed manner, 5 algorithms to improve the Free Distance of PCSRCCs.
3. This small depth length allows to extensively test the faithful of results.

Extensive results with size 105 are given in part (4), but the algorithm can be used with larger depths.

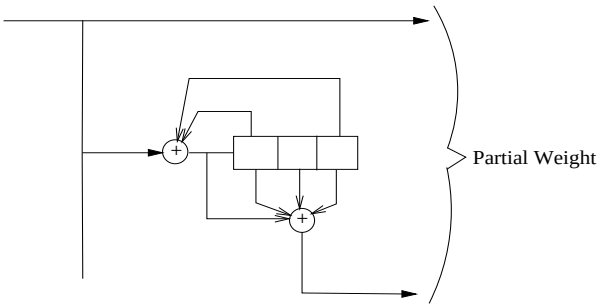

Figure 7 Partial weight of the coder

3.1 The Search Space

The search space is the space of all possible interleaving matrices of depth 105. So, a population element is an interleaving matrix. To generate them, we consider any permutation matrix of depth N as the product of (N-1) transpositions.

3.2 The Test space

As stated in (Seghers 1995, Benedetto and Montorsi 1996*b*, Maseng 1997, Lance C. Perez and Jr. 1996), the Free Distance of a PCSRCC code is led by self-terminating vectors of low weight. So, the test space is the space of the self-terminating vectors of weight less than 6. We will first define the *partial weight* of a vector for a given coder.

Définition 3.1 (Partial Weight) *The partial weight of a vector for a given coder, is the sum of the systematic part weight and the weight obtained by coding the vector with the first coder (equivalent to a* $\frac{1}{2}$ *systematic recursive convolutional encoder).*

An example of "partial weight" is given on figure (7). First a list of self-terminating vectors of weight less than 6, having a "1" in first position is and a partial weight less than $(X-2)$ (where X is an upper bound of the estimated achievable Free Distance) is generated. $X-2$ is chosen because the minimum weight led by a vector with weight different from zero to a recursive convolutional coder is $\omega_{min} = 2$ (Benedetto and Montorsi 1996*a*). In part (4), $X = 23$; as a solution better than 23 is never found, a larger value for X is useless. Each vector is translated for as many positions as possible to the right, and the list is ordered according to the increasing partial weight.

3.2.1 Crossover

It is difficult to find a good crossover operator with interleaving matrices and it is also very time-consuming for the computer. Moreover, results were not improved by the use of crossover at the cost of a very larger complexity in the program. Thus the crossover was not used ($P_c = 0$).

3.2.2 Mutation

A random position of a worst-vector is chosen, and this position is changed with a random position in the matrix. This is illustrated on figure (8). Mutation is applied to half the population ($P_m = 0.5$).

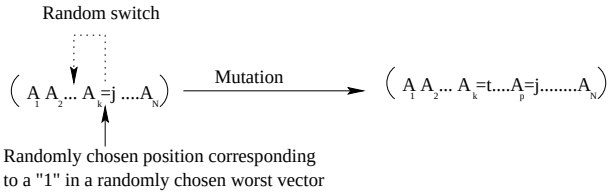

Figure 8 Mutation Principle

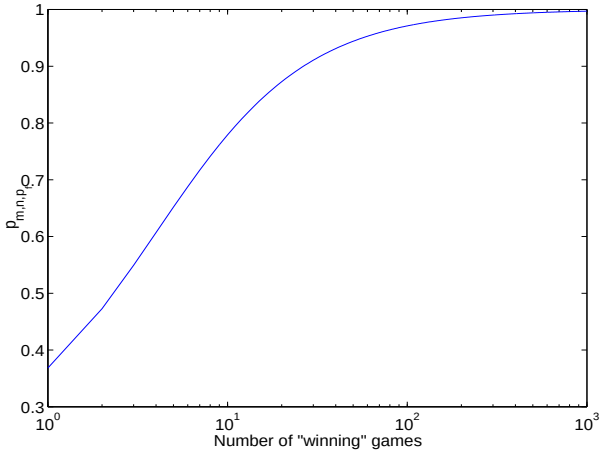

Figure 9 *Reliability* **table with** $p = 0.95$

3.3 Evaluation Function

The most simple and logical way to associate a value to each matrix, telling how "good" the matrix is, seems to be to associate to it the worst distance it had led to until the current generation. This criteria does not take into account the ancientness of a matrice, and maybe a very young population element population would kick out an elder one which would have been evaluated over almost all the search space. For instance, a matrix with $d_{free} = 18$ that has already been evaluated over 1.000.000 vectors can be better than a matrix with $d_{free} = 19$ but which has only been evaluated over 1.000 vectors. In order to consider this problem, the concept of *reliability* (as defined in (Alliot 1995) for optimizing an Othello program) was used. In order to evaluate an Othello program, the number of victories against a reference program is used. The reliability of a program depends on this number, but also on the number of games already played. A program that won 46 games out of 48 may be more reliable than one that won 6 games out of 6. If p is the probability to win a game, the probability to win m games out of n is given by:

$$P(p,m,n) = \left(\begin{array}{c} n \\ m \end{array} \right) p^m (1-p)^{(n-m)} \qquad (2)$$

Let's now find the value of p according to the number of winning games m over the number of played games n. A confidence rate $p_f \in [0,1]$ is chosen. It can be then shown that for $p_f \in [0,1]$, p is higher than a certain value p_{m,n,p_f} with probability p_f, with the following implicit definition of p_{m,n,p_f}:

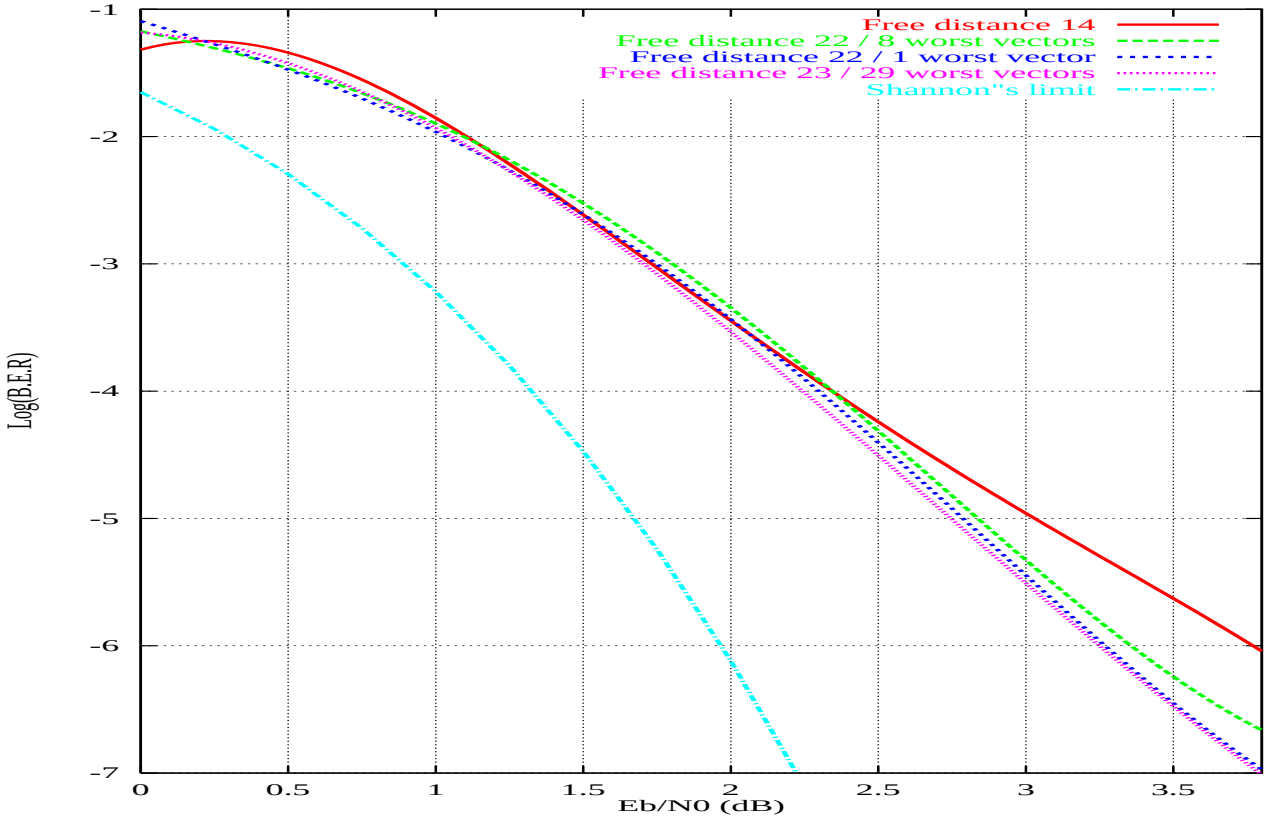

Figure 10 Comparison in terms of B.E.R. of different matrices

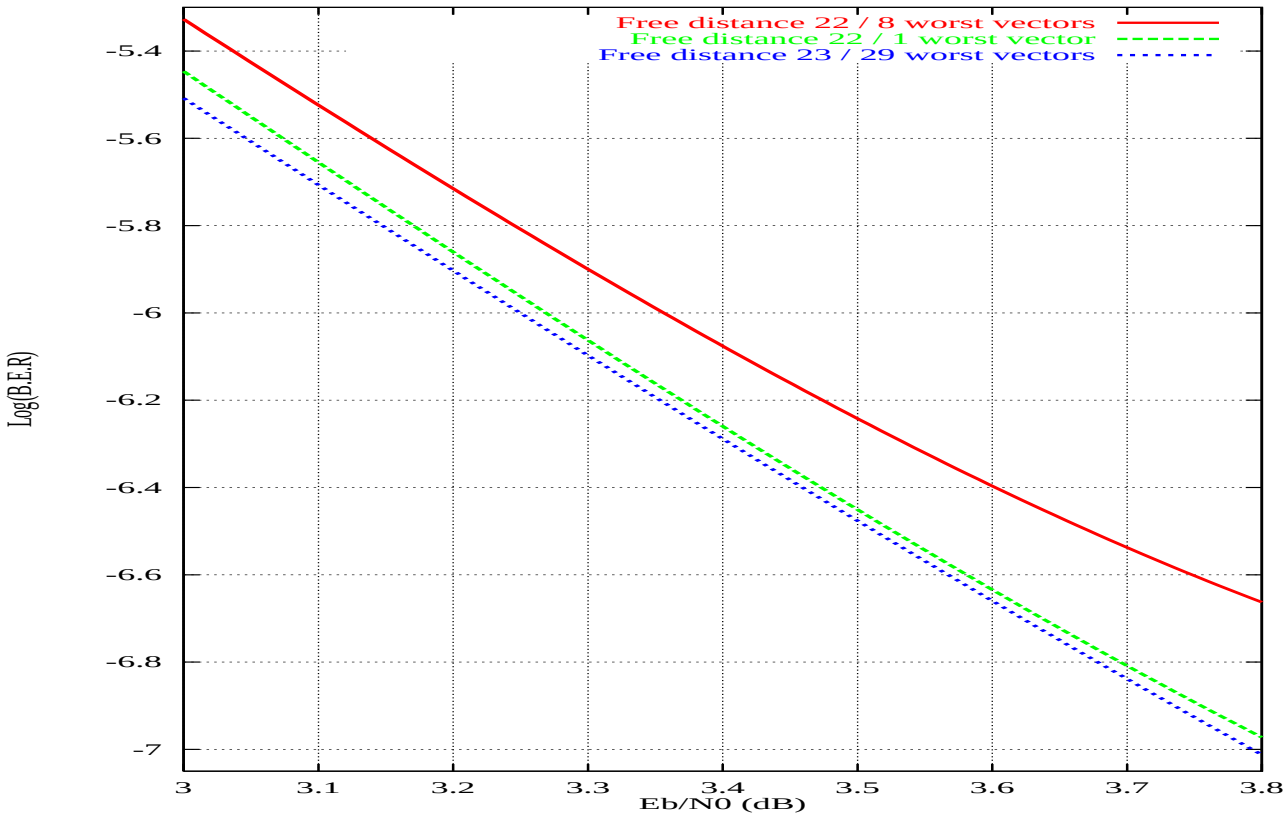

Figure 11 Zoomed comparison in terms of B.E.R. of different matrices

$$\int_{p_{m,n,p_f}}^{1} P(p,m,n)dp = \frac{p_f}{n+1} \quad (3)$$

In part (4), $p_f = 0.95$, figure (9) gives the values of p_{n,n,p_f} for different values of n.

The vocabulary used in (Alliot 1995) was adapted: a winning game will correspond to X "good" evaluations for a matrix. This means that, for instance, if a matrix is evaluated over Y vectors from the sample group during a generation and, if its minimum distance is better than the current worst one, this matrix will add $\frac{Y}{X}$ more winning games at this generation. The winning games are added to the "surviving" matrices throughout the generations.

At each generation, a matrix is tested over 1000 ($X = 1000$) vectors of the list and also over every vector that has decreased the Free Distance of an "ancient" element (a matrice which has been tested over 5000 times and led to a Free Distance greater than 19 before it was "broken"). Actually, this has been a great improvement: vectors that have "broken" a former good matrix are kept and first tested on new generated matrices. Indeed, as new matrices are generated by mutation, they have common characteristics and the same vector often "breaks" lots of matrices. Consequently, the number of tests operated on the matrices increases as generations go on: for instance, in the following results, it starts at 1000 and ends at about 10000.

The final fitness function is:

$$f = (D_f + \frac{1}{1+N_w}) \times p(N_g)$$

where D_f is the minimal distance found, N_w is the number of worst vectors corresponding to the minimal distance and N_g is the number of games played.

This method is empirical, and the underlying idea is to allow the population to get old (in order to have strong matrices) slowly (in order to allow new matrices, maybe better than the elder ones, to arise).

3.4 General Remarks

In the results presented in part (4), an unpunctured code is used (its rate is $\frac{1}{3}$). Afterwards, as all the worst vectors are kept for the final matrix, an adapted puncturing scheme can be found in order to have the same Free Distance but a higher coding rate. The program was stopped after 24 hours.

4 Results

4.1 105 bits sized problem

The coder described on figure (4) and the Forward-Backward algorithm with no simplification for the Turbo Decoding algorithm are used. Matrices with d_{free} =19 are found in about 3 minutes (on a Pentium pro 300), with d_{free} =20 in about 10 minutes, with d_{free} =21 in about 20 minutes, with d_{free} =22 in about 40 minutes and finally with d_{free} =23 in 6 hours. Our first remark is that in 40 minutes, a better result (because of a lower number of worst vectors) than the best interleaving matrices for PCSRCCs (Maseng 1997) (which was been obtained in one week) is obtained. Second, the algorithm can be applied to larger sizes of interleavers.

The efficiency (in terms of B.E.R.) of different matrices is given on figure (10). The highest curve shows a randomly chosen interleaving matrix. Its Free Distance is 14. The second curve shows the performance of the best matrix found in (Maseng 1997), which has a Free Distance of 22 and 8 worst vectors. The third curve corresponds to one of the matrices found with the genetic algorithm, with a Free Distance of 22 but only 1 worst vector. The influence of the number of worst vectors for the same Free Distance can be seen: the less worst vectors, the better the perfomance. The fourth curve of figure (10), shows the performance of a matrix leading to a Free Distance of 23 with 29 worst vectors. This is the best matrix obtained (see figure (11)). Indeed, it has the greater Free Distance, but also a good spectrum (at least until a weight of 29). The last curve is Shannon's asympotic limit.

Table (1) gives the spectrum of vectors for different Free Distance optimization algorithm (Maseng 1997). The two last lines correspond to the results obtained with genetic algorithms.

Genetic algorithms can improve substantially the performances compared to those of classical existing methods. Extension of the method to larger size problems is discussed in the last part.

4.2 Extension to larger sizes

The method discribed in this article can be adapted to larger problems. However, the set of self-terminating vectors increases quickly. Indeed, a self-terminating vector can be a combination of 2 self-terminating vectors of size 2 and 2, or 2 and 3, or 3 and 2 or 2 and 4 or 4 and 2 or 3 and 3, or even of 3 self-terminating vectors of size 2. For example, if the size of the problem is 105, there are 28951 self-terminating vectors with a 1 at the first position and a partial cost lower than 21. If we delete the combinations of 2 or more self-terminating vectors, only 7713 *simple* self-terminationg vectors remain. For a problem of size 256, the number of self-terminating vectors increases to 66097 whereas we still have 7713 simple self-terminating vectors. Consequently, for larger size problems, self-terminating vectors can be built by dynamically combining self-terminating vectors. First trials are being performed giving Free Distances of 26 for the 256 bits size problem described earlier. Simulations are still being performed and the size of the problem will be increased to 512, 1024 and 2048 in a near future.

5 Conclusion

Improving the *effective Free Distance* to lower the error floor as proposed by (Benedetto and Montorsi 1996*a*) does not work, at least on our example. But, the *Free Distance* and the distances spectrum seems to be a good criteria to optimize. It

Hamming distance	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
1. Block (5x21)	1	3	1	2	4	6	2	2	5	12	11	16	191	45	123	74	268	301	474	603
2. Pseudo rand (105 bits)	-	1	0	0	3	2	4	9	7	13	11	22	41	36	52	94	170	204	328	466
3. Non uniform (10x10)	-	-	-	-	1	0	1	8	2	7	1	31	55	31	29	69	155	181	304	409
4. Block helical simile (5x21)	-	-	-	-	-	-	-	-	-	24	0	0	71	206	36	183	169	132	905	574
5. Hokfelt & al.	-	-	-	-	-	-	-	-	-	-	-	-	8	20	36	87	122	177	241	286
6. Genetic algorithm 1	-	-	-	-	-	-	-	-	-	-	-	-	1	46	52	74	125	148	246	306
7. Genetic algorithm 2	-	-	-	-	-	-	-	-	-	-	-	-	-	29	52	69	118	151	233	312

Table 1 Matrices spectrum for $N = 105$

was observed that the higher the Free Distance, the better performance and for equal Free Distances, the less worst vectors, the better performances.

Applying genetic algorithms to difficult information theory optimization problems seems to be a very promising direction. In the case of our concern, genetic algorithms have proved their efficiency by giving the best published results on an interleaving depth of 105 bits. Moreover, this algorithm is not limited by the size and higher interleaver sizes are already being tested.

References

Alliot, Jean-Marc (1995). A genetic algorithm to improve an othello program. In: *Proceedings of EA95*. Springer Verlag.

Benedetto, S. and G. Montorsi (1996*a*). Design of parallel concatenated convolutional codes. *IEEE Transactions on Communications* **44**(5), 591 – 600.

Benedetto, S. and G. Montorsi (1996*b*). Unveiling turbo codes: Some results on parallel concatenated coding schemes. *IEEE Transactions on Information Theory* **42**(2), 409 – 428.

C.Berrou, A. Glavieux and Punya Thitimajshima (1993). Near shannon limit error-correcting coding and decoding: turbo-codes. In: *ICC'93*. Geneva. pp. 1064 – 1070.

Goldberg, D.E (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading MA Addison Wesley.

Lance C. Perez, Jan Seghers and D. J. Costello Jr. (1996). A distance spectrum interpretation of turbo codes. *IEEE Transactions on Information Theory* **42**(6), 1698 – 1709.

Maseng, Johan Hokfelt & Torleiv (1997). Methodical interleaver design for turbo codes. In: *International Symposium on Turbo Codes*. Brest, France.

Michalewicz, Z (1992). *Genetic algorithms + Data Structures = Evolution Programs*. Springer-verlag.

Robertson, Patrick (1994). Illuminating the structure of decoders for parallel concatenated recursive systematic (turbo) codes. In: *IEEE Globecom Conference*. pp. 1298 – 1303.

Seghers, J. (1995). On the free distance of turbo codes and related product codes. Diploma project SS 1995 n0. 6613. Swiss Federal Institute of Technology.