

Discovering Efficient Turbo Code Interleavers by Genetic Algorithms

Pavel Krömer¹, Václav Snášel¹, Nabil Ouddane²

¹Department of Computer Science, Faculty of Electrical Engineering and Computer Science
VŠB – Technical University of Ostrava,
17. listopadu 15, 708 33 Ostrava – Poruba, Czech Republic
{pavel.krömer.fei, vaclav.snasel}@vsb.cz

²Department of Telecommunications, Faculty of Electrical Engineering and Computer
Science, VŠB – Technical University of Ostrava,
17. listopadu 15, 708 33 Ostrava – Poruba, Czech Republic
nabil.ouddane.st1@vsb.cz

Abstract. Since the appearance in 1993, first approaching the Shannon limit, the Turbo Codes give a new direction for the channel encoding field, especially since they were adopted for multiple norms of telecommunications, such as deeper communication. To obtain an excellent performance it is necessary to design robust turbo code interleaver. We are investigating genetic algorithms as a promising optimization method to find good performing interleaver for the large frame sizes. In this paper, we present our work, compare with several previous approaches and present experimental results.

Keywords: Genetic Algorithms, Turbo Codes interleaver optimization

1 Introduction

It is known that the encoding block on the transmission scheme is one of the complex operations. Channel encoding adds redundancy symbols to the message to be transmitted causing diminution to spectral efficacy of the transmission. The Turbo Codes (TC) was a new tendency in the channel encoding field and they have become a reference soon after their introduction. Their original name comes from their first structure introduced and described in [1], namely concatenated convolutive recursive systematic codes with iterative decoding.

Turbo codes offer the best compromise between structure (concatenation) and randomness created by the interleaver. Its characteristic iterative decoding process is among the principal performance factors of the turbo codes. The significant characteristics of turbo codes are small bit error rate (BER) achieved even at low signal to noise ratio (E_b/N_0) and the error floor at moderate and high values of E_b/N_0 . Previous studies proved that the random interleaver can be in certain cases more efficient than other channel encoding schemes [2].

In this paper, genetically evolved interleaver will be compared to random interleaver by the means of BER to evaluate its efficiency. The increase of the interleaver size gives better performance and better interleaving gain while worsening latency. The relation (1) illustrates the influence on the latency

$$t_d = \frac{K_f}{R_b} N_i \quad (1)$$

where R_b is the code bit rate, K_f stands for the frame size and N_i is the number of the decoding stages. The performance of the turbo codes depends on two principal parameters, first is the code spectrum, and the second is decorrelation between the external information at the same number of iterations.

The optimization can be used for the amelioration of performance and the diminution of the matrix stature with safe performance. This second point is very interesting for multimedia real-time transmission systems over satellite because the interleaving matrix makes a considerable diminution of the codec complexity and delay. Interleaver matrix sizes vary from tens to ten-thousands of bits. It is highly inefficient to test all the possible input vectors (2^N) with all the possible interleaver matrices ($N!$), requiring $2^N \cdot N!$ tests. Therefore, advanced interleaver optimization methods are required.

Some results based on genetic algorithm exploitation, were previously obtained in [3] but only for small value of N ($N=50$), achieving a performance gain of 0.1 dB. Recent researches [4] focused on larger frame sizes and it means also larger interleaver length N , which is more promising for real time communication over satellite. The choice of the objective function was based on the maximizing of the determinist performance parameter which is free distance. Such optimization is complicated due the fact that free distance calculation is non-trivial and complex task.

In following, we are using similar parameters and compare our results to interleavers discovered by the means of previous methods. In the next section is presented short overview of a turbo code system.

1.1 Turbo Code System

Figure 1a represents convolutive encoder used in the past experiences, with the dimension of the memory effect $v = 4$, constraint length $L = (v + 1) = 5$ and

rate $r = \frac{k}{n} = \frac{\text{InputSymbols}}{\text{OutputSymbols}} = \frac{1}{2}$. The encoder is punctured for rate of 1/3 to get

better maximal free distance.

The turbo encoder presented in Figure 1b, is the same as first turbo encoder used in [5]. It is composed of parallel concatenation of two convolutive systematic recursive codes connected with an interleaver in between. The rate of such encoder is 1/3.

The first encoder operates directly on the input sequence, denoted by c , of length N . The first component encoder has two outputs. The first output, denoted by v_0 , is equal to the input sequence since the encoder is systematic. The other output is the

parity check sequence, denoted by v_1 . The interleaved information sequence at the input of the second encoder is denoted by c' .

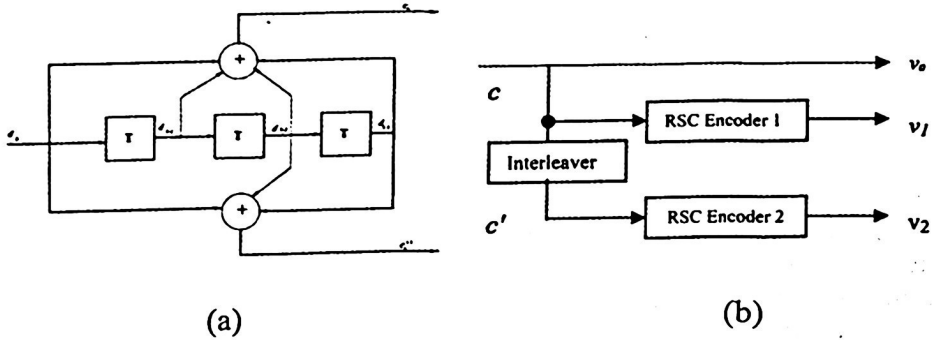


Fig. 1. Convolutional encoder (a) and turbo encoder (b)

Only the parity check sequence of the second encoder, denoted by v_2 , is transmitted. The information sequence v_0 and the parity check sequences of the two encoders v_1 and v_2 , are multiplexed to generate the turbo code sequence. The overall code rate is thus 1/3.

In this article will be used the Enhanced Maximum A Posteriori Probability Log Map soft decoding algorithm, a technique often used for the satellite communication decoding. More details can be found in [6].

2 Evolutionary Algorithms

Evolutionary algorithms (EA) are family of iterative stochastic search and optimization methods based on mimicking successful optimization strategies observed in nature [7, 8, 9, 10]. The essence of EAs lies in the emulation of Darwinian evolution utilizing the concepts of Mendelian inheritance for the use in computer science and applications [10]. Together with fuzzy sets, neural networks and fractals, evolutionary algorithms are among the fundamental members of the class of soft computing methods.

EA operate with population (also known as pool) of artificial individuals (referred often as items or chromosomes) encoding possible problem solutions. Encoded individuals are evaluated using objective function which assigns a fitness value to each individual. Fitness value represents the quality (ranking) of each individual as solution of given problem. Competing individuals search the problem domain towards optimal solution [8]. In the following sections will be introduced general principles common for all methods belonging to the class of evolutionary algorithms.

2.1 Evolutionary Search Process

For the purpose of EAs is necessary proper encoding, representing solutions of given problem as encoded chromosomes suitable for evolutionary search process. Finding proper encoding is non-trivial problem dependent task affecting the performance and results of evolutionary search while solving given problem. The solutions might be encoded into binary strings, real vectors or more complex, often tree-like, hierarchical structures, depending on the needs of particular application area.

The iterative phase of evolutionary search process starts with an initial population of individuals that can be generated randomly or seeded with potentially good solutions. Artificial evolution consists of iterative application of genetic operators, introducing to the algorithm evolutionary principles such as inheritance, survival of the fittest and random perturbations. Current population of problem solutions is modified with the aim to form new and hopefully better population to be used in next generation. Iterative evolution of problem solutions ends after satisfying specified termination criteria and especially the criterion of finding optimal solution. After terminating the search process, evolution winner is decoded and presented as the most optimal solution found.

2.2 Genetic Operators

Genetic operators and termination criteria are the most influential parameters of every evolutionary algorithm. All below presented operators have several implementations performing differently in various application areas.

Selection operator is used for selecting chromosomes from population. Through this operator, selection pressure is applied on the population of solutions with the aim to pick more promising solutions to form following generation. Selected chromosomes are usually called parents.

Crossover operator modifies the selected chromosomes from one population to the next by exchanging one or more of their subparts. Crossover is used for emulating sexual reproduction of diploid organisms with the aim to inherit and increase the good properties of parents for offspring chromosomes.

Mutation operator introduces random perturbation in chromosome structure; it is used for changing chromosomes randomly and introducing new genetic material into the population.

Besides genetic operators, termination criteria are important factor affecting the search process. Widely used termination criteria are i.e.:

- Reaching optimal solution (which is often hard to recognize)
- Processing certain number of generations
- Processing certain number of generations without improvement in population

EAs are successful general adaptable concept with good results in many areas. The class of evolutionary techniques consists of more particular algorithms having numerous variants, forged and tuned for specific problem domains. The family of evolutionary algorithms consists of genetic algorithms, genetic programming, evolutionary strategies and evolutionary programming.

2.3 Genetic Algorithms

Genetic Algorithms (GA) introduced by Holland and extended by Goldberg are wide applied and highly successful EA variant. Basic workflow of originally proposed standard generational GA is:

-
- I. Define objective function
 - II. Encode initial population of possible solutions as fixed length binary strings and evaluate chromosomes in initial population using objective function
 - III. Create new population (evolutionary search for better solutions):
 - a. Select suitable chromosomes for reproduction (parents)
 - b. Apply crossover operator on parents with respect to crossover probability to produce new chromosomes (offspring)
 - c. Apply mutation operator on offspring chromosomes with respect to mutation probability. Add newly constituted chromosomes to new population
 - d. Until the size of new population is smaller than size of current population go back to a.
 - e. Replace current population by new population
 - IV. Evaluate current population using objective function
 - V. Check termination criteria; if not satisfied go back to III.
-

Many variants of standard generational GA have been proposed. The differences are mostly in particular selection, crossover, mutation and replacement strategy [8].

3 Designing Genetic Algorithms for Interleaver Optimization

Genetic algorithms have been already used for interleaver matrix optimization. Durand et al. [4] used customized GA to optimize the interleaver of the size 105, comparing their results to previous interleaver design techniques. Their genetic algorithm was fully based on mutation and the crossover operator was due to complications omitted. The fitness criterion for every interleaver was maximum free distance.

Rekh et al. [3] presented recently another GA for the interleaver optimization, introducing 2-point crossover to interleaver evolution process. Nevertheless, the crossover impact is limited by necessary correction of errors created during crossover application. The fitness criterion was BER and the size of optimized interleaver 50.

In the following section will be discussed our GA for interleaver optimization and compared to previous mentioned approaches.

3.1 Interleaver GA Discussion

An interleaver of the dimension N performs a permutation of N input bits and therefore can be seen as a general permutation of N symbols. Hence, we encode interleaver for the purpose of genetic algorithm as permutation σ_N . An interleaver of the dimension N performs a permutation of N input bits and therefore can be seen as a general permutation of N symbols $\sigma_N = (i_1, i_2, \dots, i_N)$, where $i_k \in [1, N]$ and $i_m \neq i_n$ for

all $m \neq n \in [1, M]$. The application of σ_N on input vector I_N for $N = 5$ is shown in relation (2).

$$\begin{aligned} I_5 &= (0, 1, 0, 1, 1) \\ \sigma_5 &= (5, 3, 4, 1, 2) \\ O_5 &= \sigma_5(I_5) = (1, 0, 1, 0, 1) \end{aligned} \quad (2)$$

The same encoding was used also in [3]. Durand et al. in [4] do not specify their interleaver encoding, although we can conclude that they used similar interleaver representation.

We have used two types of selection: roulette wheel selection and for speeding up the convergence of the algorithm a semi-elitary hybrid selection scheme choosing one parent by elitary manners and the second by proportional manners of roulette wheel selection. Mutation is simply realized by swapping positions of two coordinates in σ_N . On the contrary, traditional crossover operators (except of uniform crossover) will corrupt the structure of permutation σ_N and hence cannot be used without some post processing used for chromosome fixing. Authors of [4] have fully omitted crossover and the crossover application in [3] lead to the need to repair every new chromosome created via crossover. This is a remarkable fact since crossover is referred as the primary operator for GA [9].

To enable the application of crossover for interleaver optimization, expecting performance increase, we have investigated the effect of uniform crossover on convergence ability of the classical interleaver optimizing GA. In the second phase, we have designed modified GA allowing the use of virtually any crossover operator for permutation evolution without corrupting the chromosomes. New crossover friendly GA is based on separation of chromosomes into groups of the same size called higher level chromosomes, (HLC's). Genetic operators are then applied on HLC's while original chromosomes act as genes as shown in Figure 2. We have tested above introduced techniques on a benchmarking problem consisting of search for an identity matrix. The results have revealed that GAs with semi-elitary selection and HLC were most efficient.

The best performing GA was used for interleaver optimization. As fitness criterion was adopted approach introduced in [3]: average BER, captured after simulated transmission of several low weight information frames.

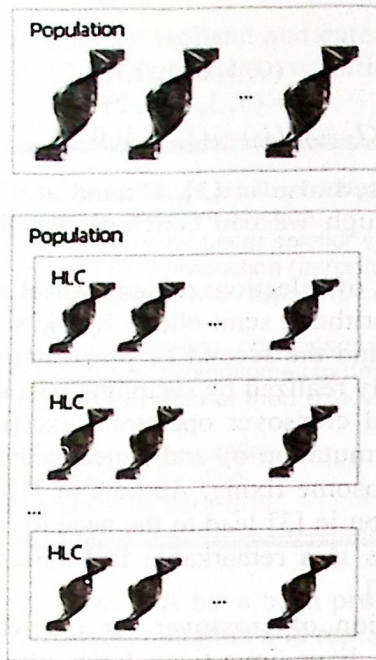


Fig. 2. Traditional population compared with HLC's

4 Experiments

An experimental framework built upon the IT++ library¹ was used to experimentally evaluate proposed interleaver generation method. We have experimented with 64, 128, 512 and 1024 bit interleavers aiming to optimize in the future as large interleaver as possible.

The settings for all optimization experiments were:

- GA with HLC and semi-elitary selection
- 100 generations
- probability of crossover 0.8
- population of 5 high level chromosomes per 6 genes
- fitness criterion was minimal BER after simulated submission of 100 random frames of weight up to 6
- the simulations were performed over additive white Gaussian noise (AWGN) channel

The AWGN channel is a good model for satellite and deep space communication links but not an appropriate model for terrestrial links. The evolved interleavers were

¹ IT++ is available at <http://itpp.sourceforge.net/>

evaluated by simulated transmission over AWGN channel for $E_b/N_0 \in [0, 4]$ and flat Rayleigh fading channel for $E_b/N_0 \in [0, 6]$. Rayleigh fading channel was used as a reasonable model for tropospheric and ionospheric signal propagation as well as the effect of heavily built-up urban environments on radio signals [11].

In the experiments, GA with classic population and semi-elitary selection were used. Optimized interleavers were compared to random interleaver taken as reference by the means of E_b/N_0 to BER ratio, captured after simulated transmission of 500 random frames.

4.1 Optimization Results

Optimization results are summarized in Figures 3 and 4. In all of them is used following notation: curve denoted as *O1* corresponds to the best interleaver found by GA with classic population, *O2* describes performance of best interleaver found using GA with HLCs and *Rand* denotes reference random interleaver. *AWGN* denoted curves illustrate experiments over additive white Gaussian noise channel and *Rayleigh* curves represent the experimental results measured over Rayleigh channel. In all figures can be seen that optimized interleavers perform better than reference random interleaver.

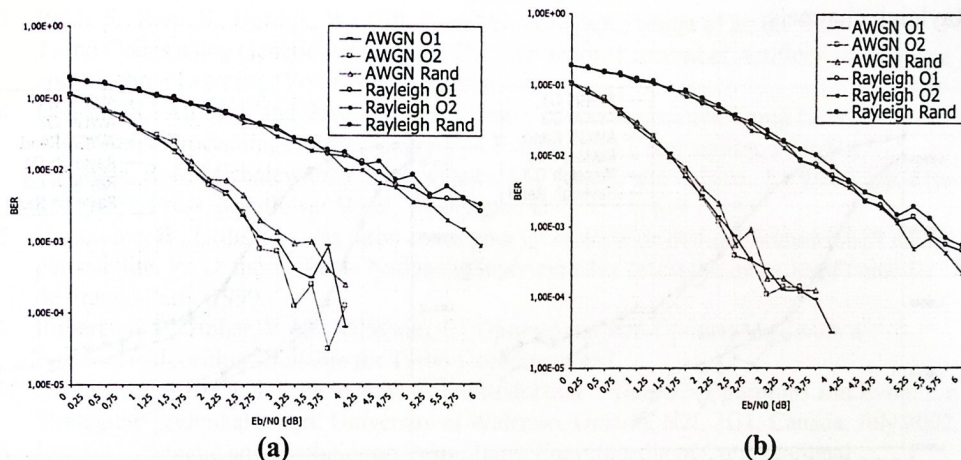


Fig. 3. 64bit (a) and 128bit (b) interleaver

5 Result Discussion

Figure 3a illustrates the binary error rate for an interleaver with the length of 64 bits; we can observe that an improvement for *AWGN* channel begin to appear from $E_b/N_0 = 2$ dB and becomes more significant for larger E_b/N_0 values, especially for the interleaver obtained by second optimization method. Both optimized interleavers

overperformed the random interleaver. For $\text{BER}=10^{-3}$ we have an E_b/N_0 of approximately 3.25 dB for the random interleaver and 2.75 dB for the second optimized interleaver achieving gain of 0.5 dB. The trend is valid for Rayleigh channel experiments as well and the supremacy of interleaver *O2* is even more evident.

For 128 bits interleaver length, we can observe, as shown on Figure 3b, that for *AWGN*, the amelioration begins to be significant between the second optimization and the random interleaver from $E_b/N_0=2.25$ dB, it means for a larger signal noise rate values. For $\text{BER}=10^{-3}$ we have E_b/N_0 equal 2.25 dB for the second optimization and 2.5 dB for the random interleaver having 0.25 dB of gain. For the Rayleigh channel transmissions, the better performance of *O1* and *O2* comparing to random interleaver becomes to be clear for greater E_b/N_0 values (>5 dB) and *O2* is again best performing among the three.

The gain becomes more considerable for interleaver length of 512bits as shown on Figure 4a, for example in *AWGN*, we have for $\text{BER}=10^{-4}$ the $E_b/N_0=2.75$ dB for the second optimization method while having 3.5 dB for the random interleaver. This indicates 0.75 dB gain for 512bits length interleaver, which is for this interleaver length a remarkable result. In Rayleigh channel, the initial BER values are for all three compared interleavers almost the same while for higher E_b/N_0 , interleaver *O2* achieves permanent gain over similarly performing *O1* and random interleaver.

Similarly, optimized 1024 bit interleavers, highlighting specially interleaver *O2*, shown at Figure 4b outperform reference random interleaver for both, *AWGN* and Rayleigh channels.

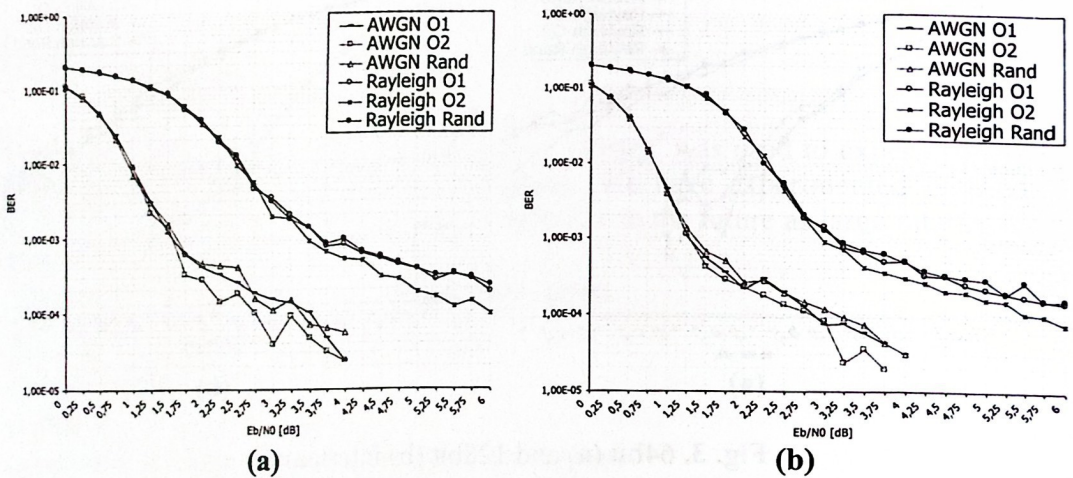


Fig. 4. 512bit (a) and 1024bit (b) interleaver

6 Conclusions

In this paper, we discussed the problem of efficient turbo code interleaver optimization by the means of genetic algorithms. Previous approaches were revised and novel modifications to existing interleaver optimizing GA improving their

convergence were introduced. Presented GA modifications are general and can be used in other application areas as well. Optimized interleavers found by introduced method were verified and compared to random interleaver by the means of BER performance. The verification utilized both, *AWGN* channel and more real-life-like Rayleigh fading channel. The optimized interleavers outperformed random interleavers having the interleaver found by presented method total winner by the means of BER to EbN0 ratio in most cases.

In the future we aim to use developed algorithm for the optimization of larger interleavers and investigate the use of minimum free distance as more competent fitness criterion. Additionally, we want to employ Rayleigh fading channel model at the optimization phase and compare obtained interleavers to interleavers evolved over *AWGN* channel. Moreover, we are investigating the general process of permutation evolution since it has numerous applications in computer science (i.e. in data compression).

References

- 1 Berrou, C., Glavieux, A. and Thitimajshima, P., Near Shannon limit error-correcting coding and decoding: turbo codes, Proc. Int. Conf. on Commun., pp. 1064–1070, 1993.
- 2 Hokfelt, J. and Maseng, T., METHODOICAL INTERLEAVER DESIGN FOR TURBO CODES, International Symposium on Turbo Codes.
- 3 Rekh, S., Rani, S., Hordijk, W., Gift, P. and Shanmugam, Design of an Interleaver for Turbo Codes using Genetic Algorithms, The International Journal of Artificial Intelligence and Machine Learning, Vol. 6, pp. 1–5, 2006.
- 4 Durand, N., Alliot, J. and Bartolomé, B., Turbo Codes Optimization Using Genetic Algorithms, Proceedings of the Congress on Evolutionary Computation, edited by Angeline, P. J., Michalewicz, Z., Schoenauer, M., Yao, X. and Zalzal, A., Vol. 2, pp. 816–822, IEEE Press, Mayflower Hotel, Washington D.C., USA, 6-9 1999.
- 5 Bartolome, B., Utilisation des turbo codes pour un systeme de communications multimédia par satellite, Ph.D. thesis, Ecole Nationale Supérieure des Télécommunications France-Ile de France-Paris, 1999.
- 6 Robertson, P., Hoher, P. and Villebrun, E., Optimal and Sub-Optimal Maximum a Posteriori Algorithms Suitable for Turbo Decoding, 1997.
- 7 Dianati, M., Song, I. and Treiber, M., An Introduction to Genetic Algorithms and Evolution Strategies, Technical report, University of Waterloo, Ontario, N2L 3G1, Canada, July 2002.
- 8 Jones, G., Genetic and Evolutionary Algorithms, Encyclopedia of Computational Chemistry, edited by von Rague, P., John Wiley and Sons, 1998.
- 9 Mitchell, M., An Introduction to Genetic Algorithms, MIT Press, Cambridge, MA, 1996.
- 10 Bodenhofer, U., Genetic Algorithms: Theory and Applications, Lecture notes, Fuzzy Logic Laboratorium Linz-Hagenberg, Winter 2003/2004.
- 11 Proakis, J. G., Digital Communications, McGraw-Hill, New York, 4th edn., 2001.