World Academy of Science, Engineering and Technology
International Journal of Electronics and Communication Engineering
Vol:5, No:4, 2011

Metaheuristic Algorithms for Decoding Binary Linear Codes

Hassan Berbia, Faissal Elbouanani, Rahal Romadi, Mostafa Belkasmi

Abstract—This paper introduces two decoders for binary linear codes based on Metaheuristics. The first one uses a genetic algorithm and the second is based on a combination genetic algorithm with a feed forward neural network. The decoder based on the genetic algorithms (DAG) applied to BCH and convolutional codes give good performances compared to Chase-2 and Viterbi algorithm respectively and reach the performances of the OSD-3 for some Residue Quadratic (RQ) codes. This algorithm is less complex for linear block codes of large block length; furthermore their performances can be improved by tuning the decoder's parameters, in particular the number of individuals by population and the number of generations. In the second algorithm, the search space, in contrast to DAG which was limited to the code word space, now covers the whole binary vector space. It tries to elude a great number of coding operations by using a neural network. This reduces greatly the complexity of the decoder while maintaining comparable performances.

Keywords—Block code, decoding, methaheuristic, genetic algorithm, neural network

I. Introduction

THE current large development and deployment of wireless and digital communication encourages the research activities in the domain of error correcting codes. Codes are used to improve the reliability of data transmitted over communication channels susceptible to noise. Coding techniques create codewords by adding redundant information to the user information vectors. Decoding algorithms try to find the most likely transmitted codeword related to the received one as depicted in Fig.1. Decoding algorithms are classified into two Categories: Hard decision and Soft decision algorithms. Hard decision algorithms work on a binary form of the received information. In contrast, soft decision algorithms work directly on the received symbols [1].

Soft-decision decoding is an NP-hard problem and was approached in different ways. Recently artificial intelligence techniques were introduced to solve this problem. Among the related works, the decoding of linear block codes using algorithm A* [9], genetic algorithms [10],[11] and neural networks [12].

Genetic Algorithms are search algorithms that were inspired by the mechanism of natural selection where stronger individuals are likely the winners in a competing environment. They combine survival of the fittest among string structures with a structured yet randomized information exchange to form a search algorithm with some of the innovative flair of human search. In each generation, a new set of artificial creatures (chromosomes) is created using bits and pieces of the fittest of the old [10],[13].

M. Berbia, Elbouanani and Romadi are now assistant professors with the Department of Networks and Communication at ENSIAS-Rabat, Morocco
M. Belkasmi is now professor with the same Department at ENSIAS-Rabat, Morocco

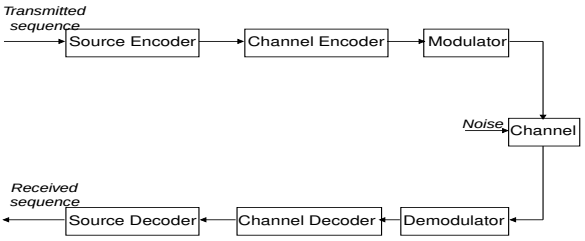

Fig. 1. Communication system model

A artificial neural network, or supply feed forward neural network, is an interconnected group of artificial neurons that uses a mathematical model for information processing, based on the parallel architecture of animal brains [10].

In this paper, we introduce two decoders based on genetic algorithms. The second decoder is improved by the use of neural networks. These decoders can be applied to any arbitrary binary linear code, in particular for codes with unknown algebraic decoder. In order to show the effectiveness of this decoder we applied it on some BCH, QR and convolutional codes.

This paper is organized as follows. Section 2 describes the decoding of linear block codes based on genetic algorithms. The neural network enhanced the decoder will be presented in section 3. Section 4 reports the simulation results and discussions. Finally, Section 5 presents the conclusion and future trends.

II. The first decoder

Let C denote a (n, k, d) binary linear code of generator matrix G, and let $(r_i)_{1 \leq i \leq n}$ be the received sequence over a

communication channel with noise variance $\sigma^2 = \frac{N_0}{2}$ where N_0 is noise power spectral density (W/Hz).

Let N_i, N_e and N_g denote respectively the population size, the number of elite members and the number of generations.

Let p_c and p_m be the crossover and the mutation rates.

A. Decoding Algorithm

The decoding based genetic algorithm is depicted on Fig 2. The steps of the decoder are as follows:

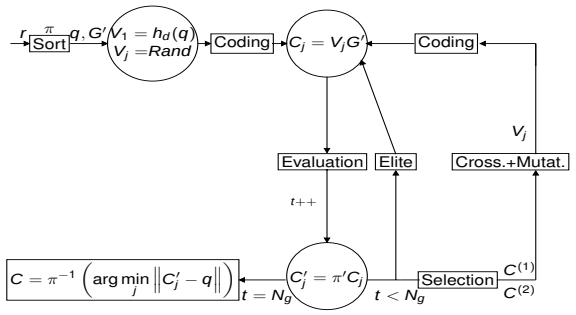

Fig. 2 Decoding of Linear Block Codes based genetic algorithm

Step 1. Sorting the sequence r in descending order ($q = \pi r$), such that the first k columns of the matrix $G' = \pi G$ are linearly independent :

$$\begin{cases} q_i = r_{p_i} \quad 1 \le i \le n, \quad p_i \in \{1, .., n\} \\ |q_1| \ \ge \ |q_2| \ \ge \ \cdots \ \ge \ |q_n| \end{cases}$$

Step 2. Generate an *initial* population of N_i binary vectors of k bits:

2.1. The first member V_1 of this population is obtained by the quantization of the q:

$$V_{1i} = \begin{cases} 1 \ \textit{if} \ q_i > 0 \\ 0 \ \textit{otherwise} \end{cases}, \quad i \le k$$

2.2. The other $N_i - 1$ members $(V_j)_{2 \le j \le N_i}$ are uniformly-random generated.

2.3. Encoding the N_i members using the new matrix G' :

$$C_j = V_j G', \qquad 1 \le j \le N_i \tag{1}$$

Step 3. For i from 1 to N_g

3.1. The population is sorted in ascending order, $C'_j = \pi' C_j$, of member's fitness defined as the distance between the modulated individual $\overline{C}_j = 2C_j - 1$ and q :

$$f_j = \sum_{k=1}^{n} (\overline{C}_{jk} - q_k)^2, \qquad 1 \le j \le N_i$$

To reduce the computational complexity of this expression, and since $\sum_{k=1}^{n} q_k^2$ is constant, and $\overline{C}_{jk}^2 = 1$, the fitness can be simplified as :

$$f_j = -\sum_{k=1}^{n} \overline{C}_{jk} q_k \tag{2}$$

3.2. The first (elite) N_e best members of this generation are inserted in the next one.

3.3. The other $N_i - N_e$ members V_j of the next generation as generated as follows :

a. Selection operation : Select pairs of individuals $(C'^{(1)}, C'^{(2)})$ use the following linear ranking method is

$$w_j = w_{\max} - \frac{2(j-1)\,(w_{\max} - 1)}{N_i - 1}, \quad 1 \le j \le N_i \tag{3}$$

where w_j is the jth member weight and $w_{\max} = 1.1$, is the the maximum weight associated to the first member.

b. Crossover operator : Create a new vector V_j *"child"* of k bits. Let *Rand* be a uniformly random value between 0 and 1 generated at each occurrence. The crossover operator is defined as follows :

If $Rand_1 < p_c$, then the ith bit of child $(V_j)_{N_e+1 \le j \le N_i}$ $(1 \le i \le k)$ is given by :

$$V_{ji} = \begin{cases} C'^{(1)}_i \ \textit{if} \ C'^{(1)}_i = C'^{(2)}_i \\ \textit{otherwise} \begin{cases} C'^{(1)}_i \ \textit{if} \ Rand_2^{(i)} < p \\ C'^{(2)}_i \ \textit{else} \end{cases} \end{cases} \tag{4}$$

where

$$p = \begin{cases} \frac{1}{1+e^{-I_i^{(s)}}} \ \textit{if} \ C'^{(1)}_i = 1 \ \textit{and} \ C'^{(2)}_i = 0 \\ \frac{e^{-I_i^{(s)}}}{1+e^{-I_i^{(s)}}} \ \textit{if} \ C'^{(1)}_i = 0 \ \textit{and} \ C'^{(2)}_i = 1 \end{cases} \tag{5}$$

and $I_i^{(s)}$ denote the systematic information of the ith symbol. For AWGN channel, $I_i^{(s)} = \frac{4q_j}{N_0}$.

It is clear that if the ith bit of the parent are different, then for greater positive value q_j, the function $\frac{1}{1+e^{-I_i^{(s)}}}$ converge to 1. Furthermore, the ith bit of child has a great probability to be 1.

Note that if $Rand_1 \ge p_c$:

$$V_j = \begin{cases} C'^{(1)} \ \textit{if} \ Rand < \frac{1}{2} \\ C'^{(2)} \ \textit{else} \end{cases} \tag{6}$$

c. Mutation operator :

If the crossover operation realized, the bits V_{ji} are *muted* with the mutation rate p_m :

$$V_{ji} \leftarrow 1 - V_{ji} \quad \textit{if} \ Rand_3^{(i)} < p_m \tag{7}$$

3.4. Encoding the N_i members of the next generation using the new matrix G' :

Step 4. The best member from the last generation is returned as the decoder decision.

B. Computational complexity

In [6], it is shown that the computational complexity of the first algorithm is polynomial and is less than that of OSD and Chase-2 algorithm for great values of code block length. This make our first decoder more efficient in term of performance and complexity

III. The Second Decoder

In this section, we present the second decoder for linear block codes, and based on genetic algorithm, and neural networks (NDAG) [17]. In contrast to the previous algorithm, the whole vector space is $\{0,1\}^n$. The individuals encoding operations and searching the systematic generator matrix of equivalent code are then ignored. Furthermore, the computational complexity of this second decoder is reduced.

The most differences with the previous decoder can be described as follows :

- The individuals of the population are not necessary the codewords,
- NDAG doesn't sort the received sequence r in descendent order of symbols reliability $|r_i|$ for searching the equivalent generator matrix G' *of the original code*,
- NDAG can be applied, in *general*, to a code generated with it generator polynomial g, and not necessary to a code identified by it generator matrix G. Furthermore, it complexity is reduced again,
- The crossover and mutation operations are applied on vectors of length n (both systematic and redundancy bits) and not only to the first k bits,
- The fitness is corrected by penalty value, introduced to favorite the vectors near the codewords. Then, the far is the Hamming distance of the individual from a codeword, the great is the penalty value :

$$fitness = \|r - Indiv.\| + \underbrace{p.Weight(Error\_class(Indiv.))}_{\text{Penalty}} \tag{8}$$

where $\|.\|$is the Euclidean distance, the *class representant (coset leader) is the most likely error vector, and* p is the penalty coefficient.

The Multilayer Perceptron (MLP), presented in Fig. 3 is used to classify the individuals of the genetic algorithm population according to their distance from the nearest codeword. It maps the vector to a penalty value.

Cybenko [14], [15] has proved that a MLP of N_i inputs, $(t+1)$ output where t is the correction capacity of code,

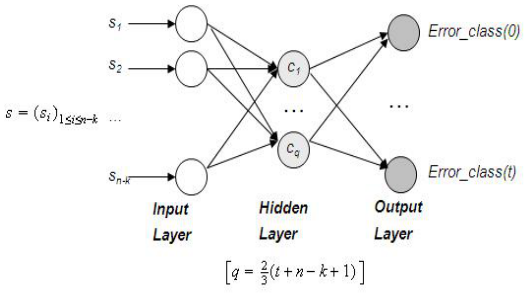

Fig. 3 Multilayer Perceptron classifier

and one hidden layer of approximately $\frac{2}{3}(n-k+t+1)$ neurons, can approxime any nonlinear continuous function f. In general, the function f is in general monotonic increasing, which transforms the individual at input to the most likely error vector at the output.

The fitness used in the first decoder was based only on the Euclidean distance between a code word and the received word and is no more sufficient for this work. We have enhanced our decoder with a multilayer perceptron which calculate a penalty value that is used to correct the fitness value.

A. Decoding Algorithm

The first step of NDAG algorithm, presented in Fig. 4, is to do an learning, for each code and regardless of Signal to Noise Ratio (SNR), using the backpropagation algorithm with learning rate $\eta = 0,5$ and sigmoid function. For each individual V_j of n bits, we compute the associated syndrome $s^{(j)} = V_j H^t$ to deduce the error vector of weight less than t. This is repeated until the stability of both matrices $(w_{ij})_{\substack{1\le i\le n-k\\ 1\le j\le q}}$ et $(v_{ij})_{\substack{1\le i\le q\\ 0\le j\le t}}$.

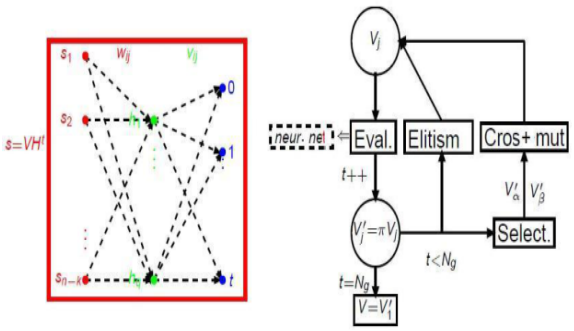

Fig. 4 The NDAG algorithm

Then an initial population of N_i information vectors $(V_j)_{1\le j\le N_i}$ of n bits is generated and then evaluated using the

neural network using the new fitness (8). Sort by increasing order of fitness of individuals will copy the N_e elites in the new generation. By applying the biased roulette wheel selection with linear classification (3) based on the fitness (8), we select a pair of individuals to cross (4) or (6) giving birth to a child. After changing the bits of the child (7), the individual is copied into the new generation.

The selection, crossover and mutation steps are repeated until the next generation is completed, and the production process of generations is repeated until the latest generation. The word decided is the first individual of the last generation.

IV. Simulation Results

The results correspond to simulations of a transmission system using a Additive White Gaussian Noise (AWGN) channel and BPSK modulation. Before studying the performance of DAG on the block and convolutional codes, a preliminary step was to optimize p_c, p_m, N_g, N_i et $w_{\max}$. Expected the Fig. 5 and the Figures of RSC simulations, the performances of DAG were obtained for $p_c = 0.97$, $p_m = 0.03$, $N_g = 100$, $N_i = 300$ et $N_e = 1$. Stopping the simulation corresponds to a minimum of 30 erroneous blocs. The performance is given in terms of BER as a function of Energy to Noise Ratio $\frac{E_b}{N_0}$.

A. DAG Results

a) Effect of individuals number: The expansion of the overall number of individuals $N_g.N_i$, will generally expand the search space of code words closest to r, which improves the performance of the DAG. The Fig. 5 shows the effect of parameters N_i and N_g on DAG for $RQ(103, 52, 19)$ code. The gain curve associated with $N_g.N_i = 30000$ compared to that using 1000 individuals is $1.4dB$ at $BER = 10^{-4}$.

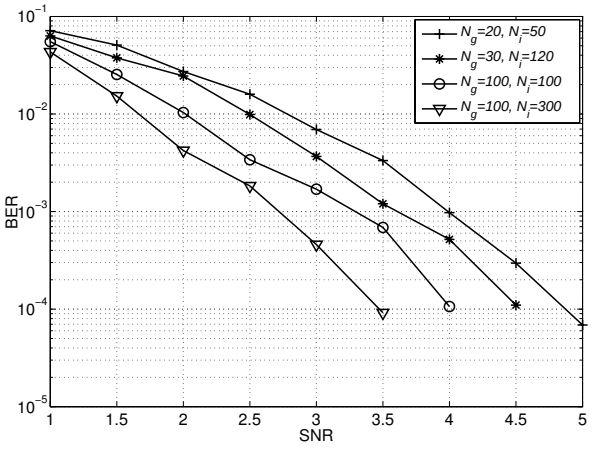

Fig. 5 Effect of the individuals and generations numbers on DAG applied to $RQ(103, 52, 19)$

b) Effect of code length: The Fig. 6 compares the DAG performances of five codes of rate $\simeq \frac{1}{2}$. Except the small code, all other codes have performance almost equal and this is possibly due to the parameters of DAG, in particular, the number of individuals chosen fixed for all codes. Indeed, when the code dimension k increases, this number covers a small percentage of the search space. Which may decrease the chances of finding the code word closest, especially for small $\frac{E_b}{N_0}$.

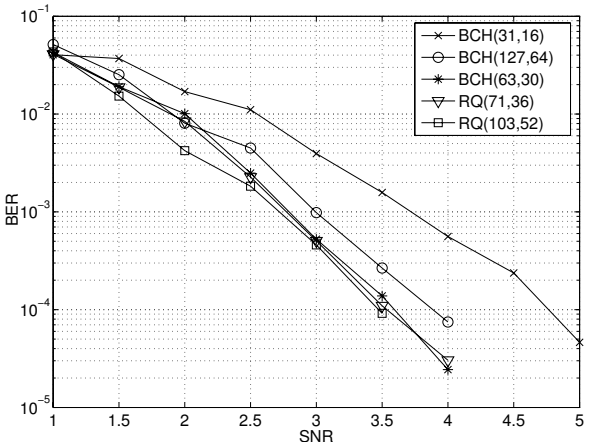

Fig. 6 Effect of length code on DAG

c) BCH(127,64,21) performance: The Fig. 7 compares the three decoding algorithms applied to the $BCH(127, 64, 21)$ code; it shows that the DAG is better than Chase-2 and the $OSD(1)$. Despite the large number of test sequences used by Chase-2 (2^{10} sequences), the DAG algorithm exceeds the performance level being less complex. An increase in the order of the OSD can exceed the performance of the DAG on the entire range of SNR. The gain of the decoder based on DAG over the $OSD(1)$ at $BER = 2.10^{-5}$ is $1dB$.

d) RQ(71,36,11) performance: The fig. 8 shows that DAG reached almost OSD-3 for the $RQ(71, 36, 11)$ code. Increasing the individuals number $N_g.N_i$ would exceed the OSD-3. However, the complexity of the DAG is less than that of OSD-3.

e) Convolutional codes Performances: Consider a Recursive Systematic Convolutional code (RSC) of length n, dimension k, memory $m \in \{2, 3, 4\}$, and generator g such as $k + m = \frac{n}{2}$. Since DAG complexity is independent m as shown in [6], Our basic idea was to see if DAG performance exceeds that of Viterbi Algorithm (VA) to conclude that DAG is effective both in terms of complexity and reliability [16].

DAG optimization began by identifying the best crossover and mutation rates for a large number of individuals. Then, for these two parameters found, we decrease the number $N_g N_i$

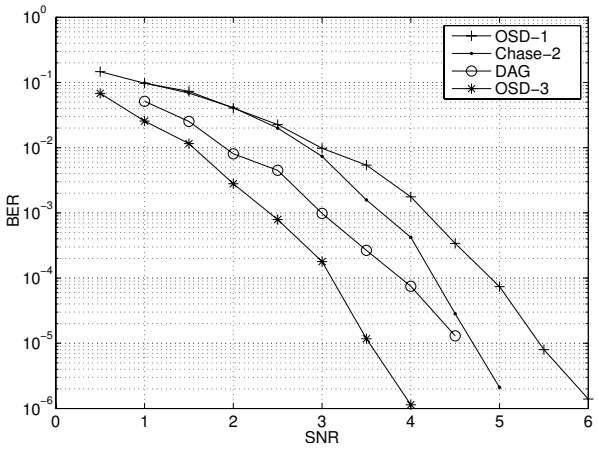

Fig. 7 Performances of DAG, Chase-2 and OSD for $BCH(127, 64, 21)$

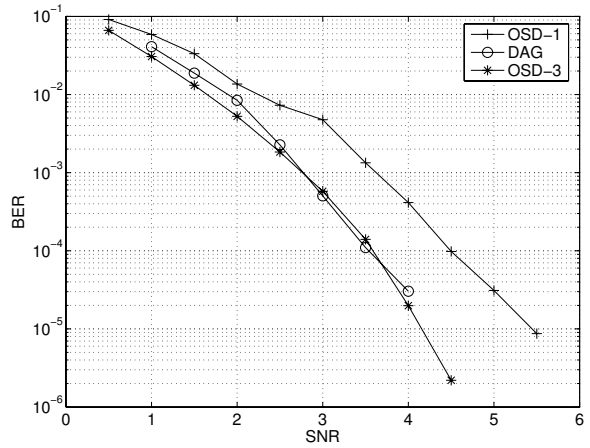

Fig. 8 Performances of DAG and OSD for $RQ(71, 36, 11)$

until the performance starts to degrade to take *optimal* values.

Unless explicitly stated, the table I lists the default values of channel, simulation and DAG parameters optimized for RSC$\big(4, \frac{1}{2}\big)$ of memory $m = 3$, with non-coded blocks of length 57, and coded blocks of length 120.

Table I
DEFAULT PARAMETERS OF DAG SIMULATION FOR RSC

p_c	p_m	N_g	N_i	N_e
0.7	0.01	50	100	1
Modul.	Channel	Err. Min.	Min. of Block	Min. erron. frames
BPSK	AWGN	200	1000	50

As for the table II, it presents the three codes RSC of rate $\frac{1}{2}$ decoded by DAG :

a. *Optimal crossover rate*

Table II
RSC DECODED BY DAG

M	2	3	4
n	120	120	120
k	58	117	116
G	$(7, 5)_8$	$(13, 15)_8$	$(21, 37)_8$

The first Fig. 9 shows, for $p_m = 0.03$, the p_c effect on the DAG performances for RSC$\big(4, \frac{1}{2}\big)$. The value 0.8 eis the best value found. However, we chose 0.7 for two reasons :

- The gain between the two curves 0.7 and 0.8 is very small at à 10^{-5} (around $0.08dB$) ,
- The smaller is the probability p_c, the DAG complexity decreases (calculations of expressions will be ignored (7)).

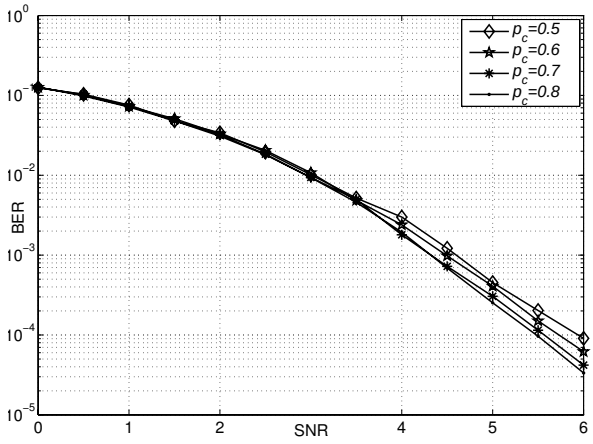

Fig. 9 Effect of p on DAG for RSC$\big(\frac{1}{2}\big)$ with bloc length equal to 120

b. *Optimal mutation rate*

For the optimal value found of p_c, we would have made the search for the best mutation probability. Fig. 10 shows the effect of p_c on DAG where the values 0.01 and 0.02 are the best. For the same reason indicated above, we chose the smallest value (ignore the expression calculation (7)).

c. *Effect of RSC memory*

Generally, the RSC is better as far as his memory is great. Fig. 11 shows this fact by simulating the three RSC$\big(5, \frac{56}{120}\big)$, RSC$\big(4, \frac{57}{120}\big)$ and RSC$\big(3, \frac{58}{120}\big)$ for $p_c = 0.7$ and $p_m = 0.03$. The gain between $m = 2$ and $m = 3$ is $0.5dB$ at 10^{-5}.

B. NDAG performance

The default settings for the simulation with NDAG, including the penalty factor p are shown on the table III.

a. *Effect of p_c and p_m*

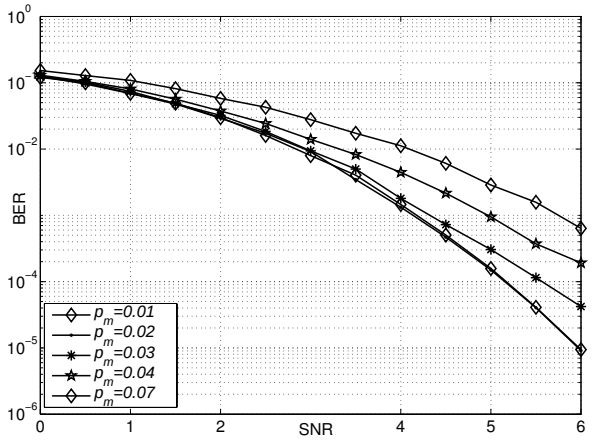

Fig. 10 Effect of p_m on DAG of Fig. 9.

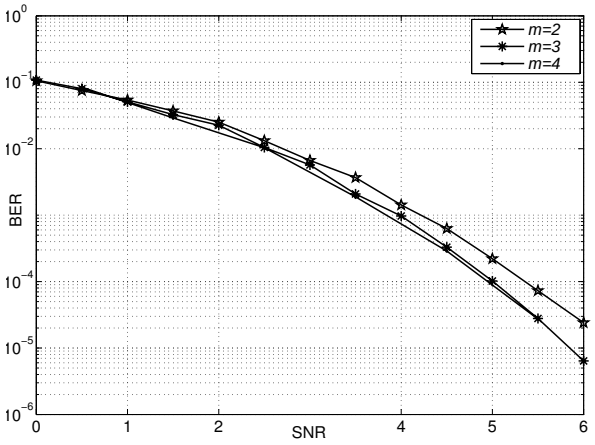

Fig. 11 Effect of memory m of RSC on DAG

Fig. 12 shows that p_c has almost no influence on the performance of the decoder for $BCH(31, 21, 5)$. One possible reason is the choice of penalty p. Also note that it is desirable to choose for p_c the smallest value 0.6 to reduce the execution time of NDAG.

In contrast, Fig. 13 shows improved performances, for the same code, for values of the probability p_m near 0.

b. *Effect of penalty coefficient*

The considerable effect on the coefficient p, on the perfect binary Golay code $(23, 12, 7)$ is depicted in Fig. 14. Theoretically, the coefficient p is strongly related to the algorithm convergence speed. If p is large, the algorithm becomes inefficient in terms of acceleration, and if this coefficient is small, NDAG promote the *wrong* solutions. For that, this value, which possibly depends on the capacity of correction, must be well optimized by simulation for large SNR and for each code.

Table III
DEFAULT PARAMETERS OF NDAG SIMULATION

p_c	p_m	N_g	N_i	p	Min. rror	Min. # of blocks
0.7	0.03	100	500	2	100	1000

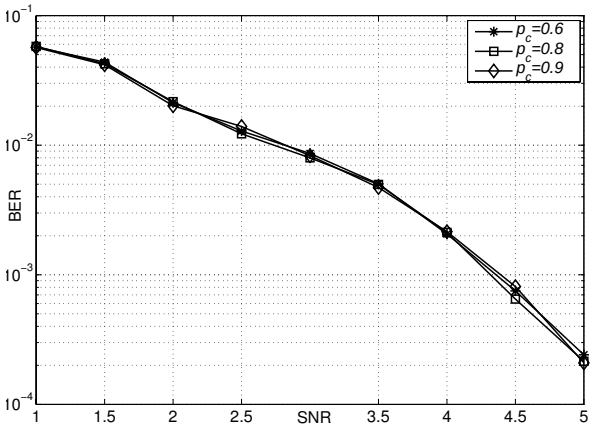

Fig. 12 Effect of p_c on NDAG for $BCH(31, 21, 5)$

c. *Effect of initial population*

Fig. 15 shows a comparison between the performances given by a randomly generated initial population and another population where individuals are chosen from the test sequences, used in Chase-2 algorithm, closest to the received word The gain between the two curves is about $0.5dB$, and sensible to be greater for $P_e < 10^{-5}$.

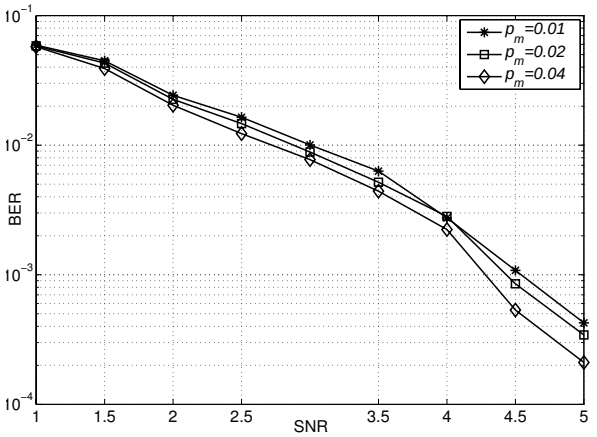

Fig. 13 Effect of p_m on NDAG for $BCH(31, 21, 5)$

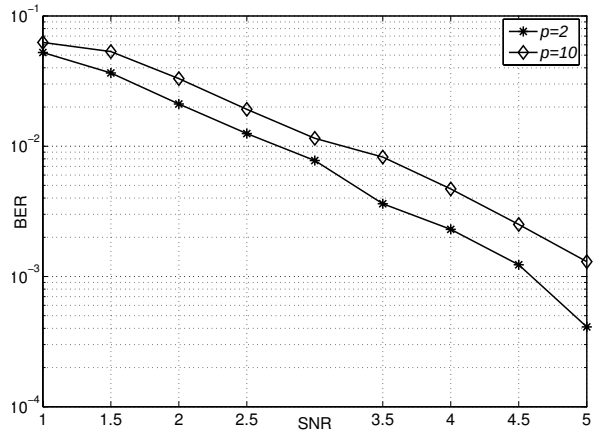

Fig. 14 Effect of penalty coefficient p on NDAG for $Golay(23, 12, 7)$

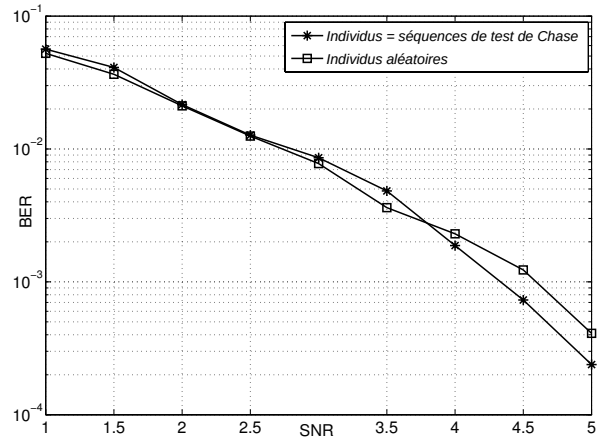

Fig. 15 Effect of individuals choice in initial population on NDAG for $BCH(31, 21, 5)$

V. Conclusion

In this paper, we have presented two decoders that are based on methaheuristics. They can be applied to any arbitrary binary code. The first algorithm is based on genetic algorithm and the second one employs both genetic algorithm and neural network to enhance performance and complexity. Our results on BCH, Convolutional, RQ and Golay codes show the effectiveness of the decoding algorithm.

The seconde decoder as well as the previous one is generic, but is less complex for polynomial codes. This work encourage the research in the field of neural network assisted genetic algorithms for the decoding problem.

References

[1] G. C. Clarck, J.B. Cain , "Error-Correction Coding for Digital Communications", New York Plenum, 1981

[2] G.D. Forney, Jr., "Generalized Minimum Distance Decoding", IEEE Transactions on Information Theory, vol. IT-12, pp. 125-131, April 1966

[3] Ja-Ling Wu, Yuen-Hsien Tseng, and Yuh-Ming Huang, "Neural Networks Decoders for Linear Block Codes", International Journal of Computational Engineering Science, vol.3, No.3, pp.235-255, 2002

[4] M.P.C. Fossorier and S. lin, "Soft decision decoding of linear block codes based on ordered statistics", IEEE Trans. information theory Vol. 41, pp. 1379-1396, sep. 1995.

[5] H. Morelos-Zaragoza, "The Art of Error Correcting Coding", Second Edition Robert , John Wiley & Sons, Ltd. ISBN: 0-470-01558-6, 2006

[6] F. El Bouanani, H. Berbia, M. Belkasmi, H. Ben-azza, "Comparaison des décodeurs de Chase, l'OSD et ceux bas´es sur les algorithmes génétiques", GRETSI 2007, Troyes, French 11-14, September 2007

[7] H. Berbia, F.El Bouanani, M.Belkasmi, F.Ayoub, R.Romadi, "Optimisation des performances d'un dcodeur a base d'algorithmes génétiques", Wotic07, Rabat 5-6 July 2007, Maroc

[8] M. Belkasmi, H. Berbia, F. El Bouanani, "Iterative decoding of product block codes based on genetic algorithms", SCC 2008, 14-16 January 2008, Ulm Germany

[9] Y. S. Han, C. R. P. Hartmann, and C.-C. Chen, "Efficient maximum-likelihood soft-decision decoding of linear block codes using algorithm A*", Technical Report SU-CIS-91-42, School of Computer and Information Science, Syracuse University, Syracuse, NY 13244, December 1991

[10] J. Holland, "Adaptation in Natural and Artificial Systems", University of Michigan Press 1975.

[11] H.S. Maini, K. G. Mehrotra,C. Mohan, S. Ranka, "Genetic Algorithms for Soft Decision Decoding of Linear Block Codes", Journal of Evolutionary Computation, Vol.2, No.2, pp.145-164, Nov.1994

[12] A.G. Scandura,A.L.Daipra,L.Arnone,L.Passoni,J.C. Moreira, "AGenetic Algorithm Based Decoder for Low Density Parity Check Codes" Latn American Applied Research 2006

[13] D. E. Goldberg, "Genetic Algorithms in search, Optimization, and machine learning", Addison-Wesly, Reading M.A, 1989.

[14] G. Cybencko ,"Approximation by superpositions of a sigmoïdal function," Mathematics of Control, signals and systems, 2, pp. 303-314, 1989.

[15] M.T. Mitchell ,"Machine learning," New York: The McGraw-Hill Companies, 1997.

[16] H. Berbia, M. Belkasmi, F. El Bouanani, F. Ayoub, "On the decoding of convolutional codes using genetic algorithms", Intern. Conf. on Computer and Commun. Engineering ICCCE'2008, pp 667-671, Malaysia, 2008

[17] H. Berbia, F.El Bouanani, M.Belkasmi, F.Ayoub, R.Romadi, "An Enhanced Genetic Algorithm Based Decoder for Linear Codes," ICTTA'08 , Damascus, Syria, 2008

Hassan Berbia Received his M.S degree in Electrical Engineering Automation and industrual Comuting. He joined IERA (Institut d'Etude et de Recherche pour l'Arabisation) in 1987. He joined Med V Suissi University-ENSIAS, Rabat (Ecole Nationale Supérieure d'Informatique et d'Analyse des Systèmes) in 2001 as assistant professor in Networking and communication department and now he is the head of the Mobile and Embedded systems Major.

Faissal El Bouanani Received his M.S degree in Network and Communication Engineering in 1996. He received his PHD in information and coding technology in 2009. He joined Med V Suissi University-ENSIAS in 2009 as assistant professor in Networking and communication department.