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, metaheuristic, 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].

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 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 $N_e$ 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:

**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:

\[
\begin{align*}
q_1 & = r_p, \quad 1 \leq i \leq n, \quad p_i \in \{1, \ldots, n\} \\
|q_1| & \geq |q_2| \geq \cdots \geq |q_n|
\end{align*}
\]

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

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

\[
V_{1i} = \begin{cases} 
1 & \text{if } q_i > 0 \\
0 & \text{otherwise}
\end{cases}, \quad i \leq k
\]

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

3. Encoding the $N_i$ members using the new matrix $G'$:

\[
C_j = V_jG', \quad 1 \leq j \leq N_i
\]

**Step 3.** For $i$ from 1 to $N_g$

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 $C_j = 2C_j - 1$ and $q$:

\[
f_j = \sum_{k=1}^{n} (C_{jk} - q_k)^2, \quad 1 \leq j \leq N_i
\]

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

\[
f_j = -\sum_{k=1}^{n} C_{jk}q_k
\]

**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 ($C_i^{(1)}$, $C_i^{(2)}$) use the following linear ranking method is

\[
w_j = w_{\text{max}} - \frac{2(j - 1)(w_{\text{max}} - 1)}{N_i - 1}, \quad 1 \leq j \leq N_i
\]

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

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

If $\text{Rand} \leq p_c$, then the $i$th bit of child ($V_j$)$_{N_i + 1 \leq j \leq N_i}$ ($1 \leq i \leq k$) is given by:

\[
V_{ji} = \begin{cases} 
C_i^{(1)} & \text{if } \text{Rand}^{(1)} < p \\
C_i^{(2)} & \text{otherwise}
\end{cases}
\]

where

\[
p = \begin{cases} 
\frac{1}{1 + e^{-I_i^{(1)}}} & \text{if } C_i^{(1)} = 1 \text{ and } C_i^{(2)} = 0 \\
\frac{e^{-I_i^{(1)}}}{1 + e^{-I_i^{(1)}}} & \text{if } C_i^{(1)} = 0 \text{ and } C_i^{(2)} = 1
\end{cases}
\]

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

It is clear that if the $i$th 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 $i$th bit of child has a great probability to be 1.

Note that if $\text{Rand} \geq p_c$:

\[
V_j = \begin{cases} 
C_i^{(1)} & \text{if } \text{Rand} < \frac{1}{2} \\
C_i^{(2)} & \text{else}
\end{cases}
\]

c. **Mutation operator**:

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

\[
V_{ji} \leftarrow 1 - V_{ji} \quad \text{if } \text{Rand}^{(1)} < p_m
\]
3.4. Encoding  the \( N_t \) 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 descendant 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:

\[
\text{fitness} = \|r - \text{Indiv.}\| + p \cdot \text{Weight(\text{Error_class(Indiv.))}}
\]

where \( \|r\| \) is the Euclidean distance, the \textit{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, 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 \( s^{(j)} = V_j^T H \) to deduce the error vector of weight less than \( t \). This is repeated until the stability of both matrices \( V_{ij} \) of \( 1 \leq i \leq n-k \) and \( (v_{ij}) \) of \( 1 \leq j \leq q \).

Then an initial population of \( N_t \) information vectors \( (V_j)_{1 \leq j \leq N_t} \) 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.

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 the search space of code words closest to \( e \). The gain curve associated with \( e \) is to optimize \( p \) on the block and convolutional codes, a preliminary step and BPSK modulation. Before studying the performance of the search space of code words closest to \( e \). The gain curve associated with \( e \) is to optimize \( p \) on the block and convolutional codes, a preliminary step and BPSK modulation. Before studying the performance of

The word decided is the first individual of the last generation.

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 the search space of code words closest to \( e \). The gain curve associated with \( e \) is to optimize \( p \) on the block and convolutional codes, a preliminary step and BPSK modulation. Before studying the performance of

The word decided is the first individual of the last generation.

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 the search space of code words closest to \( e \). The gain curve associated with \( e \) is to optimize \( p \) on the block and convolutional codes, a preliminary step and BPSK modulation. Before studying the performance of

The word decided is the first individual of the last generation.

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 the search space of code words closest to \( e \). The gain curve associated with \( e \) is to optimize \( p \) on the block and convolutional codes, a preliminary step and BPSK modulation. Before studying the performance of

The word decided is the first individual of the last generation.

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 the search space of code words closest to \( e \). The gain curve associated with \( e \) is to optimize \( p \) on the block and convolutional codes, a preliminary step and BPSK modulation. Before studying the performance of

The word decided is the first individual of the last generation.

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 the search space of code words closest to \( e \). The gain curve associated with \( e \) is to optimize \( p \) on the block and convolutional codes, a preliminary step and BPSK modulation. Before studying the performance of

The word decided is the first individual of the last generation.

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 the search space of code words closest to \( e \). The gain curve associated with \( e \) is to optimize \( p \) on the block and convolutional codes, a preliminary step and BPSK modulation. Before studying the performance of

The word decided is the first individual of the last generation.

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 the search space of code words closest to \( e \). The gain curve associated with \( e \) is to optimize \( p \) on the block and convolutional codes, a preliminary step and BPSK modulation. Before studying the performance of

The word decided is the first individual of the last generation.

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 the search space of code words closest to \( e \). The gain curve associated with \( e \) is to optimize \( p \) on the block and convolutional codes, a preliminary step and BPSK modulation. Before studying the performance of

The word decided is the first individual of the last generation.

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 the search space of code words closest to \( e \). The gain curve associated with \( e \) is to optimize \( p \) on the block and convolutional codes, a preliminary step and BPSK modulation. Before studying the performance of

The word decided is the first individual of the last generation.

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 the search space of code words closest to \( e \). The gain curve associated with \( e \) is to optimize \( p \) on the block and convolutional codes, a preliminary step and BPSK modulation. Before studying the performance of

The word decided is the first individual of the last generation.

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 the search space of code words closest to \( e \). The gain curve associated with \( e \) is to optimize \( p \) on the block and convolutional codes, a preliminary step and BPSK modulation. Before studying the performance of

The word decided is the first individual of the last generation.

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 the search space of code words closest to \( e \). The gain curve associated with \( e \) is to optimize \( p \) on the block and convolutional codes, a preliminary step and BPSK modulation. Before studying the performance of

The word decided is the first individual of the last generation.
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\( (4, 1) \) of memory \( m = 3 \), with non-coded blocks of length 57, and coded blocks of length 120.

Table I

<table>
<thead>
<tr>
<th>( p_c )</th>
<th>( p_m )</th>
<th>( N_p )</th>
<th>( N_i )</th>
<th>( N_e )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.7</td>
<td>0.01</td>
<td>50</td>
<td>100</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Modul.</th>
<th>Channel</th>
<th>Err. Min.</th>
<th>Min. of Block</th>
<th>Min. erron. frames</th>
</tr>
</thead>
<tbody>
<tr>
<td>BPSK</td>
<td>AWGN</td>
<td>200</td>
<td>1000</td>
<td>50</td>
</tr>
</tbody>
</table>

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

a. **Optimal crossover rate**

The first Fig. 9 shows, for \( p_m = 0.03 \), the \( p_c \) effect on the DAG performances for RSC\( (4, \frac{1}{2}) \). The value 0.8 is 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 \( 10^{-5} \) (around \( 0.08dB \)),
- The smaller is the probability \( p_c \), the DAG complexity decreases (calculations of expressions will be ignored (7)).

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 \( p_m \) 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\( (5, \frac{56}{120}) \), RSC\( (4, \frac{57}{120}) \) and RSC\( (3, \frac{58}{120}) \) for \( p_c = 0.7 \) and \( p_m = 0.03 \). The gain between \( m = 2 \) and \( m = 3 \) is 0.5\( dB \) at \( 10^{-5} \).

### Table II

**RSC DECODED BY DAG**

<table>
<thead>
<tr>
<th>( M )</th>
<th>( n )</th>
<th>( k )</th>
<th>( G )</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>120</td>
<td>58</td>
<td>((7, 5))_8</td>
</tr>
<tr>
<td>3</td>
<td>120</td>
<td>117</td>
<td>((13, 15))_8</td>
</tr>
<tr>
<td>4</td>
<td>120</td>
<td>116</td>
<td>((21, 37))_8</td>
</tr>
</tbody>
</table>

**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. 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.

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 $0.5 dB$, and sensible to be greater for $P_e < 10^{-5}$.
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 second decoder as well as the previous one is generic, but is less complex for polynomial codes. This work encourages the research in the field of neural network assisted genetic algorithms for the decoding problem.

REFERENCES


Hassan Berbia Received his M.S degree in Electrical Engineering Automation and industrial Comuting. He joined IERA (Institut d’Etude et de Recherche pour l’Arabisation) in 1987. He joined Med V Suisse 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 Suisse University-ENSIAS in 2009 as assistant professor in Networking and communication department.